Virus Contact Tracing

Description: This week we follow-up on a bunch of continuing Zoom news, since Zoom appears to be poised to become the teleconferencing platform of choice for the world at large. They've made more changes, have been sued, and have been rapidly taking steps to fix their remaining problems. We have some browser news and another worrisome look into Android apps using a novel approach to quickly characterize them. We have an interesting and sad bit of miscellany, a progress report on my SpinRite work, and then we take the sort of full technical deep dive into the joint Apple/Google Contact Tracing system that our listeners have come to expect from this podcast. By the end of this podcast everyone will understand exactly what Apple and Google have done and how the system functions, in detail.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-762.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-762-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson's here. There is, of course, lots to talk about. A farewell to a guy who was very inspirational to both Steve and me in our youth, another victim of COVID-19. And then Steve will talk about Apple and Google's plan to support quarantine and tracking using your iPhone and Android. Steve breaks it down, says how it works, and explains how it can be used without invading privacy. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 762, recorded Tuesday, April 14th, 2020: Virus Contact Tracking.

It's time for Security Now!, with a sassy Steve Gibson of GRC.com.

Steve Gibson: Now, you're going to have to explain that, Leo.

Leo: I went to - so I'm not going to say what it is. But if you mistype twitter.com/sggrc, which is his Twitter handle, if you just slightly mistype it, you get somebody called Sassy GRC, who has been a member since 2012 with that account. So they're not some fly-by-night trying to steal your thunder.

Steve: And I've never been called "sassy," so we know that's not me.

Leo: Well, I think you're kind of sassy. You've been in sassy moods.

Steve: So to no one's surprise, this week we're going to take a deep technical dive into what exactly it is that Apple and Google have combined their intelligences to design.

Leo: Oh, good.

Steve: So, but we've got a bunch of stuff to talk about. We're going to follow up on a bunch of continuing Zoom news, since Zoom appears to be poised to become the teleconferencing platform of choice for the world. They've made more changes. They've been sued, of course. And they are rapidly taking steps to fix their remaining problems. So I wanted, since that was Zoom Go Boom was last week's topic, I wanted to sort of catch up on what's been happening since.

We've got some browser news. Another worrisome look, I heard you talking about Android, Leo, on MacBreak Weekly, and I couldn't agree with you more. It's just we've got some more worrisome news from taking a look at Android apps using a novel approach to quickly characterize them and what was found. We've got an interesting and sad bit of miscellany, and I know you already know what it is.

Leo: I do.

Steve: And a progress report on SpinRite which is all good news. And then we're going to take a full technical deep dive into the joint Apple/Google contact tracing system, to the level that our listeners have come to expect from this podcast. And by the end of the podcast, everyone will understand exactly what Google and Apple have done, how the systems function in detail, and what it means. And I'm expecting that we'll have some time for you and me, Leo, to talk about, sort of to put this into context in terms of abuse and bad news and so forth.

There's been a lot of misunderstanding. I'm hearing people talking about things, I mean, even Moxie, who we know is capable of understanding this, he shot off a bunch of tweets right off the bat, I'm sure before he read the spec, which were completely incorrect about some of the system's problems. It sounds like subsequently he understands how the thing works better. But anyway, we're going to have a great podcast for our listeners once again.

Leo: Yeah. He had some real concerns, but I'm glad you're going to address this because I shared them.

Steve: Yes. There's no megabytes or gigabytes of data upload/download. That's all incorrect.

Leo: Oh, good. Oh, good.

Steve: Yup.

Leo: He envisioned a DDoS attack using this.

Steve: Bottom line is these guys really nailed it as an underlying platform from which more can be built. So anyway, I think a great podcast.

Leo: And that's exactly what Rene and the team on Tuesday, you know, earlier today concluded is, look, you've got two of the best companies in the world doing this, that nothing's going to be that surprising to them. They've thought of the these already.

Steve: Yes. I thought Andy summed it up perfectly, which is he would far rather have these two companies that understand the technology and have been so much attacked in the past, rather than some random government agency saying, well, [crosstalk].

Leo: Right. We can do this. Yeah.

Steve: Yeah.

Leo: No, but also they own the platforms. And so it's only sensible that they should be the ones that come up with the API. But we'll talk about it. I'm glad you're going to talk about this because I also think it's kind of clever. I think it's very interesting.

Steve: It's really cool how it works. Our listeners are going to love the way it works from a technical standpoint. And it wouldn't do us any good to have it a year from now, which is what the government would provide.

Leo: A year.

Steve: We need it in two weeks.

Leo: Give it to Boeing. We'll have it flying in 10 years.

Steve: So our Picture of the Week, thanks to a Twitter follower, sent this. I got a kick out of this because this is something that we've all seen. In this picture we have two construction workers, one with a sledgehammer, that have just blasted, like pounded a hole in someone's wall and interrupted the family meal. Mom and Dad are both looking in horror and shock at this big hole, and the kids are hiding behind their chairs. Actually it looks like little Susie is in a wheelchair, so a little more vulnerable. And one of them, from his hard hat, is saying, "Hi, we've changed your privacy settings." Meaning, yes, the privacy policy is changed after we're already using the product.

Leo: Ain't nothin' you can do about it.

Steve: That's right.

Leo: You've got a big hole.

Steve: We decided after looking at all the information we've been collecting, we could monetize this sucker. So, yeah. Anyway, fun cartoon, and thank you for the pointer.

Okay. Some Zoom follow-ups. They rushed out, Zoom rushed out another Zoom bombing mitigation. Until these three Zoom desktop clients were updated for Linux, Mac, and Windows, they were prominently, and originally conveniently, probably, displaying the current Zoom meeting ID in each app's title bar. This behavior, which had presumably caused no trouble at all during Zoom's first nine years of life, since its birth back in 2011, had suddenly become a significant liability as new Zoomers were sharing screen shots of their Zoom meetings on social media, without stopping to consider that their meeting ID was also being shared because the screenshot contained the title bar. So, whoops.

As we just talked about last week, yes, there were some problems, some fundamental problems with Zoom. But a lot of the problems ended up just being a manifestation of the insane overnight popularity of Zoom, things that suddenly when everyone started using it and hadn't had a lot of experience using it, they were doing things that they shouldn't have been doing. So there was an across-the-board update. The meeting ID has been moved from, well, removed from the title bar and placed into a drop-down panel for the sessions info icon, which is in the top left of the Zoom app. So there's just a little "i" circled thing. You click on it, and you can get the meeting ID from there. Which in a mature app is probably where it should be since we now know that just the Zoom ID itself can be abused.

Additionally, they've made the management and presumably the need for security-related settings more obvious to a meeting's hosts. Last week's update also added a new dedicated security icon to the app's control panel, where the meeting organizer can manage all security-related settings from a single location, rather than, as they were previously, having them scattered all over the place, and they had to like bounce around among separate dialogs in order to find them all. So these new settings include all the things we talked about last week - the ability to lock meetings, to enable the waiting room, and more.

Oh, and they added another feature. Meeting hosts are now able to enable meeting rooms on the fly, even if the feature was turned off before the start of the meeting, which hadn't been possible before. That had to be set up in the original configuration of the meeting before it was started. That they fixed. And it'll also be less often needed since the update now also enables waiting rooms by default for all new meetings. And it also mandates that all new conferences be password protected. So again, I probably sounded a little upbeat about the speed with which Zoom was addressing these problems. And I continue feeling that they are being very responsible. You know, it went mainstream and lost its innocence pretty quickly.

And predictably, speaking of losing its innocence, last Tuesday one of the company's shareholders, a Michael Drieu, filed a class-action lawsuit on behalf of all other shareholders. The suit alleges that Zoom made "materially false and misleading statements" that overstated its privacy and security measures; that it engaged in deception when it claimed that its product supported end-to-end encryption - well, it does in some cases; but as we know, not with the strength that we're used to - and also alleges that Zoom only uses encryption for the transport link, allowing the service to still access user data and putting users "at an increased risk of having their personal information accessed by unauthorized parties, including Facebook." So, yeah. This is going to happen. And who knows how this will settle out. But, you know, it will.

Meanwhile, Zoom has enlisted the aid of someone who's moderately famous, and that's Alex Stamos. They reached out to Alex, who previously worked at Facebook, and before that at Yahoo, as their, well, he's not their, well, he worked there, Facebook and then

Yahoo, as the CISO, their Chief Information Security Officer. And as we know, Alex departed Facebook two years ago, back in 2018, in protest over Facebook's handling of data security practices surrounding the Cambridge Analytica fiasco and Russian interference in the 2016 U.S. presidential election. Alex had had his eye on Facebook's Russian activity since around the summer of 2016 and wanted the company to go public with his findings. But Facebook's top execs, including Zuck, refused to allow that to happen, so he said, I think this is not where I want to be.

He's now very busy being an adjunct professor at Stanford's Freeman Spogli Institute, and a visiting scholar at the Hoover Institution. And in a posting of his on Medium last Wednesday he said he felt compelled to assist Zoom even though he's consumed by other commitments. I really liked what Alex wrote, and we've got time, so I wanted to share it with our listeners.

He said in his posting on Medium: "Last week, after I posted a series of tweets discussing the security challenges for Zoom and how they could respond, I got a phone call from Eric Yuan, Zoom's founder and CEO. We talked about the significant challenges his company was facing, both in responding to an incredible growth in users, but also living up to the security expectations of the moment. He asked detailed and thoughtful questions of my experiences working at companies facing extreme crises, and I was impressed by his clear vision for Zoom as a trusted platform and his willingness to take aggressive action to get there. He asked if I would be interested in helping Zoom build up its security, privacy, and safety capabilities as an outside consultant, and I readily agreed."

And Alex said: "To be clear, I'm not an employee or an executive of Zoom, and I don't speak for the company. I have refrained from any public comment on Zoom or discussions with journalists since my call with Eric. But in the interest of transparency, I think it's important to disclose this work. I don't do a lot of consulting these days. I'm generally quite busy with my role at Stanford, and I'm proud of the work that team has been doing during this critical time for disinformation. This opportunity to consult with Zoom was too interesting to pass up, however, and I thought I would explain why I have embraced this challenge.

"First off, Zoom has gone from being a successful mid-size enterprise IT company to a critical part of the lives of hundreds of millions in the space of a few months. As my CV might suggest, I am attracted to difficult problems, and this creates some doozies. As someone who has walked through the galaxy of blinking lights and deafening whir of tens of thousands of servers carrying the sessions of millions of users, I appreciate the effort it takes to build a product that scales. To successfully scale a video-heavy platform to such size, with no appreciable downtime, in the space of weeks is literally unprecedented in the history of the Internet. It has been clear to many people who have worked on production-scale systems that something special has been happening at Zoom, and the related security challenges are fascinating.

"It's not just the technical challenges that I am interested in. In a time of global crisis, Zoom has become a critical link between co-workers, family, friends, and most importantly between teachers and students. The morning Eric called me," he said, "(and most mornings since), there were five simultaneous Zoom sessions emerging from my home, as my three kids recited the Pledge of Allegiance in their virtual morning assembly, my wife supported her middle-school students, and I participated in a morning standup with my Stanford colleagues. Like many techies, I've used Zoom professionally for a while. But I admit that there was still a bit of culture shock as my wife taped a daily calendar full of Zoom meeting codes to our eight year-old daughter's desk.

"The adaptation of a successful enterprise collaboration tool into virtual classrooms, virtual doctors' offices, and a myriad of other applications including at least one virtual

Cabinet Room, has created privacy, trust, and security challenges that no company has ever faced. As I told the computer science students in my Trust and Safety Engineering course this last quarter - the last two weeks of which were taught over, yes, Zoom - coding flaws and cryptographic issues are important, but the vast majority of real technological harm to individuals comes from people using products in a technically correct but harmful manner. Zoom has some important work to do in core application security, cryptographic design, and infrastructure security, and I'm looking forward to working with Zoom's engineering teams on those projects."

And he finishes, saying: "Still, I'm certain that the real challenge, one faced by every company trying to provide for the diverse needs of millions seeking low-friction collaboration, is how to empower one's customers without empowering those who wish to abuse them. I encourage the entire industry to use this moment to reflect on their own security practices and have honest conversations about things we could all be doing better. This is possibly the most impactful challenge faced by the tech industry in the age of COVID-19, and together we can make something positive out of these difficult times and ensure that communications are safer and more secure for all."

So anyway, I thought that was a great piece of communication. And I think it really nicely sort of frames where Zoom is. And again, I'm impressed with the idea that Eric, the Zoom CEO, would proactively reach out to Alex and say, hey, you've been in the middle of this before. Would you be willing to give us a hand?

And following up on that, last Wednesday, the day before Alex Stamos's blog posting on Medium, Zoom's CEO Eric Yuan also announced that Zoom had formed a CISO Council and an Advisory Board to collaborate and share ideas regarding how to address Zoom's current security and privacy issues. In his announcement he wrote: "I'm truly humbled that, in less than a week after announcing our 90-day plan, some of the most well-respected CISOs in the world have offered us their time and services. This includes CISOs from VMware, Netflix, Uber, Electronic Arts, HSBC, NTT Data, Procore, and Ellie Mae." He said: "The purpose of the CISO Council will be to engage with us in ongoing dialogue about privacy, security, and technology issues and best practices to share ideas and collaborate."

So anyway, I think these are all good moves. As I noted last week, no one has yet taken a deep look into their crypto, and someone needs to. What we've seen is that their ECM, their Electronic Codebook Mode of crypto, is also most certainly leaking repeating patterns from plain text. And that suggests that it really needs a closer look. If they've more deeply rolled their own crypto solutions, and if Zoom is headed toward becoming the world's teleconferencing platform of choice, then one thing Zoom's CEO ought to do, and perhaps Alex or one of the CISOs will recommend it if they haven't already, would be to open their system to an independent security audit.

There will be, oh, there were a number of stories last week about major companies already preemptively banning all use of Zoom, just as a result of the bad press that happened, SpaceX and Google among them. I understand the emotional reaction to the initial security troubles, but most of the headline-making was, as we said at the time, from high-profile Zoom bombing, which was mostly the result of lax security measures on the part of the conference organizer, you know, this was all new to everybody, or to hundreds of millions of users, at least, due to the massive rapid uptake of Zoom. And, you know, it's probably a lot of people's first experience using the platform. So it should surprise no one that some mistakes were made.

And to Zoom's credit, they've already fixed all of those defaults and done everything they can to shore things up. But I think an independent security audit is probably now one of the many things that Zoom needs. But I'm impressed by what I've continued to see. I think they continue to make a lot of the right moves.

Browser news. We've got a couple of little tidbits. On schedule, or on coronavirus revised schedule, Google released Chrome 81 last week. As expected, Chrome is becoming less tolerant, with the release of 81, Chrome has become less tolerant of mixed content images. As we know, "mixed content" means that, whereas the underlying web page is delivered over a TLS authenticated and encrypted, thus HTTPS, connection, passive images, or actually passive content, are explicitly using HTTP rather than HTTPS. And I say "explicitly" because, if an image's URL leaves off any protocol specification, that is, the HTTP or HTTPS, the browser will default to using the same protocol, HTTPS in this case, as the underlying web page.

Until Chrome 81, mixed content was being allowed because it was regarded as passive content. Images, audio, video, and object links were all considered to be passive content. That is to say, they were allowed to be pulled over HTTP; whereas scripts and iframes, things that are more active and potentially dangerous, were considered active content, so they were already being rigorously blocked from mixed content loading. But Chrome 81 changes this. From now on, or I might say unless it results in too much breakage, in which case Google could be seen to back off, Chrome will override even an explicit http:// URL protocol and actively replace it with https. And if the asset content cannot be delivered over https, well, that's just too bad. The image won't be loaded, and it will show as broken on the web page. So that's happening now, and we'll see what the downstream consequences are for that.

Maybe it's just, you know, old web pages. If the page itself was secure, and images are coming from presumably the same domain, then it's hard to see why they couldn't be secure, too. And presumably Google has already done some instrumentation and telemetry and looked to see what was going on.

Chrome 81 also brings us a new feature, and that's Web NFC support. This is a Worldwide Web Consortium, you know, W3C standard, now here in Chrome. It will allow web pages to read and write to NFC tags when they're close to the user's laptop or computer. As we know, NFC uses near field RF technology; and in fact NFC stands for Near Field Communications, which has a range down at these power levels of two to four inches, so virtually in contact with each other.

The initial release of this API supports a widely compatible universal data interchange format known as NDEF, standing for NFC Data Exchange Format. It's a lightweight binary message format which is widely cross-NFC tag compatible. At the moment, apps are able to access a device's underlying NFC API. We see this, for example, with Apple Pay on iOS. So having native NFC support in Chrome will open the opportunity for NFC to be available to a broad range of websites.

So we'll see how that goes. Who knows what use it'll be put to. But, for example, a web page that wasn't previously able to access an NFC security tag could potentially do that, whereas it wouldn't have had direct access before that. Chrome 81 also fixed 32 different security problems of varying degrees. None were critical. Three were rated high. And the rest were about half and half, split between medium and low security. So not much to see there.

We were also just last week talking about the need to update to Firefox 74.0.1, which eliminated that pair of zero-day use-after-free memory vulnerabilities that were found being deployed in the wild in targeted attacks. Now today we have Firefox 75. It fixed a few problems. And the only real thing that changed is that they've sort of upped the ante a little bit with their background telemetry collecting. They'd always been collecting telemetry, or at least have been for a while, and they have explained what and why. I've got a link to their page in the show notes, if anyone is concerned or worried.

What they said is that: "Firefox collects telemetry data by default. We collect this to help improve performance and stability of Firefox. Telemetry data is made up of two data sets: interaction data and technical data. Interaction data includes information about your interactions with Firefox to us such as number of open tabs and windows, number of web pages visited, number and type of installed Firefox add-ons, and session length, as well as Firefox features offered by Mozilla or our partners such as interaction with Firefox search features and search partner referrals. Technical data includes information about your Firefox version and language, device operating system and hardware configuration, memory, basic information about crashes and errors, outcome of automated processes like updates, safe browsing, and activations to us.

"When Firefox sends data to us, your IP address is temporarily collected as part of our server logs. IP addresses are deleted after 30 days. If you're interested, you can read more about how we handle your data in our in-depth documentation about our data platform."

And they finish, saying: "We do not know about your specific interactions with Firefox. We just know the number of tabs a user had opened and how long they were opened." And in fact if you're using Firefox, in the URL you can put about:telemetry, which will display your browser's telemetry collected information, so you can get a sense for and see what stuff is being gathered and sent back. And it's entirely possible to opt out of providing this feedback.

Firefox says: "You can opt out of sending any Firefox telemetry information at any time. If you opt out of sending telemetry data, we will also treat this as a request to delete any data we previously collected. Data will be deleted within 30 days after you opt out." So they're handling that responsibly. Again, it is on by default; but if you go under the menu under Options > Privacy & Security and then look for Firefox Data Collection & Use, you'll find a box with some checkboxes, and you just turn off the ones that you're not comfortable with.

Okay. So what's been added to Firefox 75? Apparently, and I don't know what - it'd be interesting to know, like, behind the scenes what's really going on. But they've decided that they wanted to know what a user's browser choice was, even if the user was not using Firefox, or presumably after a user has changed their default away from Firefox, or maybe is using a different browser. So Firefox 75 now contains a new separate process which is launched by the Windows Task Scheduler once a day to ping a report back to Mozilla. And apparently they actually use ICMP ping messages. You know that pings can contain a data payload. They often just contain, like very little of nothing. But you can send data, if you're not concerned about it getting there and reliability and so forth. So that's what they do.

So Mozilla explained of this change in 75, they said: "With Firefox 75 we are launching a new scheduling task for Windows that will help us understand changes in default browser settings. As with all other telemetry-related changes here at Mozilla, this scheduled task has gone through our data review, a process designed with user choice and privacy at its core." They said: "We're collecting information related to the system's current and previous default browser setting, as well as the operating system locale and version. This data cannot be associated with regular profile-based telemetry data." So it's a separate process being sent separately. "If you're interested in the schema, you can find it here."

And they have provided a link, and I've got it in the show notes. It does show you exactly the stuff that's being sent. It's an XML file, and I looked at it. It looks very innocuous. They said: "The information we collect is sent as a background telemetry ping every 24 hours. We'll respect user-configured telemetry opt-out settings by looking at the most recently used Firefox profile. We'll respect custom enterprise telemetry-related policy settings if they exist. We'll also respect policy to specifically disable this task."

So anyone, I wanted to let our users know that's going on in case anyone cares. Again, I'm sort of - I'd love to have been a fly on the wall to know why it is that they're wanting to get this, what they're interested about, what they think might be happening. But anyway, that's happening. So we're on the same page with Chrome 81 and Firefox 75. And remember that since Chrome's 81 was delayed, Google still plans to skip 82 entirely. Apparently they're like committed to this release schedule by version number. And so they plan to deliver 83 on its regular schedule. And they're still imagining that 83 will have TLS v1.0 and 1.1 again removed. Remember they put it back, or they chose not to remove it as they were previously expecting to. Now they say they're going to do that in 83. I'll be surprised if that happens this year. I think 83 is aimed at this summer. And that still seems a little soon.

So once again, Android apps in the crosshairs. This research was conducted by a team at Ohio State University, New York University, and the Helmholtz Center for information Security, known as CISPA. The paper that resulted was titled "Automatic Uncovering of Hidden Behaviors from Input Validation in Mobile Apps." It's a very clever approach. The sheer volume, as we know, of Android apps submitted to the Google Play Store means that an automated first-pass screening system is the only possible means for even trying to put a dent in the task of discovering those that might have a hidden purpose. Otherwise, and even so, it's possible to simply have malicious apps get lost in the crowd.

The abstract of their paper, I've got a link to the entire PDF in the show notes for anyone who's interested, the abstract reads: "Mobile applications (apps) have exploded in popularity, with billions of smartphone users using millions of apps available through markets such as the Google Play Store or the Apple App Store. While these apps have rich and useful functionality that is publicly exposed to end users, they also contain hidden backdoors that are not disclosed, such as backdoors and blacklists designed to block unwanted content. In this paper, we show that the input validation behavior, that is, the way the mobile apps process and respond to data entered by users, can serve as a powerful tool for uncovering such hidden functionality." This is just a very clever approach.

They said: "We therefore have developed a tool, INPUTSCOPE, that automatically detects both the execution context of user input validation and also the content involved in the validation, to automatically expose the secrets of interest. We have tested INPUTSCOPE with over 150,000 mobile apps, including popular apps from major app stores and preinstalled apps shipped on the phone, and found 12,706 mobile apps with backdoor secrets and 4,028 mobile apps containing blacklist secrets."

Okay. So they note, deeper into the paper, they note that the problems they discovered were not just theoretical. And citing from the paper, they said: "Nor are such cases theoretical. By manually examining several mobile apps" - which their INPUTSCOPE tool found as potentially bad - "we found that a popular remote control app with 10 million installs contains a master password that can unlock access even when locked remotely by the phone owner when the device is lost.

"Meanwhile, we also discovered a popular screen locker app with five million installs uses an access key to reset arbitrary users' passwords to unlock and enter the system. In addition, we also found that a live streaming app with five million installs contains an access key to enter its administrator interface, through which an attacker can reconfigure the app and unlock additional functionality. Finally, we found a popular translation app with 1 million installs containing a secret key to bypass the payment for advanced services such as removing the advertisements displayed in the app."

What they realized, these guys, was that secret backdoors or other behaviors are often accessed through the front door. So they ran their static code analysis tool INPUTSCOPE against the input-handling logic of Android apps. And they found a significant number of

designed-in misbehavior. And as they said, these were not obscure apps. They took the top 100,000 most popular apps on Google Play, the 30,000 apps preinstalled on Samsung devices - how could you have 30,000 apps preinstalled on Samsung devices? That's what the page says - and 20,000 taken from the alternative Chinese market Baidu.

The study examined two issues: what proportion of apps exhibited secret backdoors, and how they might be used or abused. So that's a total of 100,000 from Google Play, 30,000 preinstalled on Samsung devices, 20,000 taken from the equivalent Chinese market. They examined two issues: what proportion of apps exhibited secret backdoors and how they might be abused or used. Of the 150,000 total, 12,706 exhibit a range of behaviors indicating the presence of some sort of backdoor. That is to say, in the input handling logic, INPUTSCOPE found 12,706 had, like, code for special case handling of specific things - a secret access key, a master password, or secret commands - and another 4,028 that seemed to be checking user input against specific blacklisted words such as political leaders' names, incidents in the news, or racial discrimination.

Looking at the backdoors, both Google Play apps and those from alternate stores such as Baidu showed roughly the same percentage of apps falling into the backdoor category: 6.8% from Google Play, 5.3% from Baidu. Interestingly, for preinstalled bloatware apps, that is, the apps that come preinstalled on phones, the percentage showing this behavior was double the other sources, at around 16%. And this finding supports a public open letter that was sent to Sundar Pichai at Google last January by Privacy International, which was critical of the way preinstalled apps were often not scrutinized for privacy and security problems. A separate Spanish study last year documented that the provenance of preinstalled apps was often shadowy, based on commercial relationships between phone makers that the end user would not be aware of.

The team took a closer look at 30 apps, picked at random from apps with more than a million installs, the ones I was talking about, finding that one installed with the ability for someone to remotely log into its admin interface. Others could reset user passwords, bypass payment interfaces, initiate hidden behaviors using secret commands, or just stop users from accessing specific, sometimes political content.

In Sophos's coverage of this research, they wrote: "Perhaps the biggest consequence from the study is simply how many Google Play apps exhibit these behaviors. While the Play Store is large, the fact that several thousand" - okay, 12 - "from among the top 100,000 apps have hidden backdoor behavior hardly inspires confidence. And there is currently no easy way, short of the sort of weeks-long analysis carried out by the researchers using a dedicated tool, to know which apps operate this way."

Sophos's reporting concluded: "That's not so much a backdoor as a blind spot, another problem Google's sometimes chaotic Android platform could do without."

Leo: I'm not sure how I feel about this.

Steve: How so?

Leo: Well, you explain to me. So it's not at all unusual for a developer to put a backdoor of the kind that you described where I'm going to distribute this to some people. I'll give them a special code so they don't have to see the ads without paying. Or a developer, I mean, you see that all the time when you develop software, where a developer has a "god mode" that they can test, makes it easy for them to test.

Steve: Sure.

Leo: It's not at all unusual for that to happen. Is Sophos saying the very existence of those makes them vulnerable to fuzzing and other kinds of hacking?

Steve: No, because, well, for example there was an instance of a remote control app that had a remote authentication bypass secret password.

Leo: Yeah, I mean, that's obviously bad, yeah.

Steve: Yeah.

Leo: But they also refer to some apps, I mean, what percentage of these apps just have developer backdoors for bug testing and things like that?

Steve: Or like friends and family secrets.

Leo: Right. Is that inherently harmful? Are they saying that's...

Steve: No. So I agree with you. It's not the case that all of these things are bad. What their tool finds is that the apps have undocumented behavior. And without looking more closely, we need to know what that is. Or it would be nice to know what that is.

Leo: I would say almost every program in the world has undocumented behavior. That's not at all unusual. Because, I mean, remember the Windows API secrets, the secret Windows codes that Microsoft never told you? Unless they're inherently - if it makes it vulnerable somehow because they could be manipulated, that I would understand. But I don't think...

Steve: Yeah, so I guess - so certainly less than, given the population breakdown they talked about, less than those 12,000 were malicious, but there were some that were.

Leo: Well, that would be bad, obviously.

Steve: And we would hope that the majority are not.

Leo: I guess the point is - see, I don't know why they mention the ones that are not. Like, okay. You found some that are malicious. I think they inflated their numbers, is what I'm saying.

Steve: Well, and I think also they were enamored of their input.

Leo: That they could do it.

Steve: Their static analysis input tool.

Leo: Right, that they could find them, yeah.

Steve: And said, look, look at all the special casing behavior we found.

Leo: But that's just not - I just don't think that's inherently dangerous, that that, yeah, of course, good job. But that's not inherently dangerous. Some is, potentially.

Steve: Right, right. I agree with you. Bypassing the need to pay for it, nah, not so much.

Leo: Yeah, right.

Steve: And speaking of Sophos, Sandboxie has gone open source.

Leo: Oh, good. That's great.

Steve: Yeah. Yes, it is very cool. For more than 15 years Sandboxie has been a Sophos project. It's now been released into the open source community for its continued maintenance and evolution. You and I, Leo, on Security Now! Episode 172, way back then, May 27th of 2008, our episode was titled "Sandboxie," where we took a deep dive into its operation. I had apparently, looking back at the notes, I had interviewed the original author of it, and I brought everything I learned from my interview of him into that podcast and talked about it.

Leo: I remember, yeah.

Steve: It was originally intended just to sandbox web browsers. I think it was just, well, in fact it was to sandbox...

Leo: Sandbox IE; right, right.

Steve: Sandbox IE, yes. It was just meant to sandbox Internet Explorer. Then it was broadened to others. And now it's become a very capable general purpose application isolation wrapper where you can run any apps that you're uncomfortable with in the sandbox. Sandboxie at a very low level intercepts any of the Windows API hooks which would be dangerous, things like file writing, so that the app thinks it wrote where it wrote, and if it then reads it back, it comes back, it thinks from there. But it's actually sequestered that into the sandbox, and then you're later able to decide what you want to do with it. Typically you just flush anything that it did because you want safety.

Anyway, I just wanted to let everyone know it's www.sandboxie.com. It was always free, but the sense I got, I read a little bit of background, is that Sophos was thinking, you know, we've got other fish to fry now. We're busy. We're not giving this the attention that maybe it deserves. Let's just let it loose into the open source community and so it can continue to live there, which is very cool.

Leo: Yay. Yeah, you know what, more stuff that should happen to. Because if a company's not going to continue to move forward with it, give it away.

Steve: Yes, that's perfect. So Leo, I think there'll be some intersection between us on this next topic, a little bit of miscellany. I discovered Scientific American magazine...

Leo: Yes, I know where you're going.

Steve: ...in my high school years.

Leo: Me, too, yup.

Steve: It was incredibly influential for me since its science writing was pitched at just the right level. Toward the back of every issue was a monthly column called "Mathematical Games," written by, and I imagine this guy's name is familiar to us all, Martin Gardner. And he was quite influential. Wikipedia noted that Gardner's "Mathematical Games" column became the most popular feature of the magazine and was the first thing - it certainly was for me - that many readers turned to. In September of 1977, Scientific American acknowledged the prestige and popularity of Gardner's column by moving it from the back of the magazine to the very front.

Leo: The main reason I subscribed.

Steve: Yes. The column ran - and get this - from 1956. Now that was a year after I was born, and probably one or two years before you were born.

Leo: The year I was born. It was the year I was born.

Steve: Ah, okay. From '56 through 1981, with sporadic columns afterwards. And it was the first introduction of many subjects that Gardner talked about to a wider audience. And the point of this bit of history is that it was Martin Gardner's October 1970 column, published early in my sophomore year of high school, where Gardner introduced his readers to John Horton Conway's amazing Game of Life. We've spoken of Conway's Game of Life many times through the years on this podcast. Conway's creation was incredibly elegant in the simplicity of its rules and the complexity of its results. And it rather clearly divided all people who were exposed to it into two camps: those who thought it was the coolest thing they had ever seen, and those who thought that the first group might benefit from medication.

Leo: I never got over the Game of Life. Never. It's the greatest thing ever.

Steve: No. The world hasn't. Needless to say, you and I were members of the first camp. And the game at the time consumed me for a long time. Wikipedia has a terrific page about Conway's Game of Life. Anyone who's listening to this podcast who doesn't already know what we're talking about must go to the Wikipedia page.

Leo: If you don't know what a "glider" is, go now.

Steve: And that glider gun up there in the upper right, which is emitting gliders. There are some beautiful animated GIFs or GIFs, depending upon how you pronounce it. The page is beautiful. All of our desktop PCs and smartphones have implementations of Conway's Game of Life. And while you're stuck at home waiting for this pandemic to subside, you will not be bored.

Sadly, I bring all this up because last Saturday, April 11th, COVID-19 claimed the life of John Horton Conway. Wikipedia writes: "John Horton Conway was an English mathematician active in the theory of finite groups, knot theory" - that's actually a thing, knot theory - "number theory, combinatorial game theory..."

Leo: Knot theory is all about topology. That's exciting stuff.

Steve: Yeah, that's very cool stuff, "...and coding theory. He also made contributions to many branches of recreational mathematics, most notably the invention of the cellular automaton called the Game of Life. Conway spent the first half of his long career at the University of Cambridge in England and" - unfortunately, as it turns out - "the second half at Princeton University in New Jersey" - and of course we know that New Jersey is one of the spots that's had a big flare-up of COVID - "where he held the title John von Neumann Professor Emeritus. On 11 April 2020, at age 82, he died of COVID-19 at his home in New Jersey."

Leo: So sad.

Steve: So I just wanted to mention that to our listeners. Very influential. So much, I mean, the Game of Life, it's so simple, yet what it does, what it produces is fantastic. And, I mean, there's been so much work done, like it's a perfect place for a person to practice their coding skills. What is the fastest life generator I can produce? And there has been, I mean, there have been papers written about optimizing the speed at which you iterate over a group of cells which are still alive, how to not bother spending time in dead areas and predict, I mean, just really cool things. There are things that move across the grid. And of course, you know, "C" in the Game of Life is the speed of light, which in the physical universe is the fastest that anything can travel. So of course you have C equals one grid per - one cell grid per iteration is C in the Game of Life. And so you have different things that move at different percentages of C. I mean, there's a whole vocabulary. There's loaves and blinkers and gliders and spaceships.

Leo: This is all based on three rules. That's it.

Steve: Yes.

Leo: It's three rules for how these things generate.

Steve: Yes.

Leo: It's so cool. It's so cool. I just love it. Yeah. That's one of the reasons people do it because it's very easy to write a program that does it.

Steve: Yeah, yeah, yeah. I mean, yeah.

Leo: Just couldn't be easier. It's so cool.

Steve: I have a little bit of SpinRite news. I am finally making very good progress toward the next SpinRite. I had a surprising number of challenges getting to where I've finally been for the last four days. Since I plan to be spending a lot of time during this development cycle working on a combination of SpinRite 6.x releases, and also the Beyond Recall utility which will be born from this work, I wanted to set things up right. I wanted to invest in the creation of an efficient development environment so I would not be spending a lot of time in repetitive and unnecessary overhead.

Turns out that the lack of support for 16-bit code by today's 64-bit OSes, coupled with the fact that my tool chain turns out to depend upon a number of 16-bit components, was a lot more troublesome than I expected. I also needed to link the DOS, the physical DOS testing target machines, which only know about SMBv1, that's the latest version of file sharing that - remember it was Windows for Workgroups is what Microsoft called it. I needed to link them to file shares on a Win10 machine. At my location with Lorrie I have Windows 10. Where I am here in my Fortress of Solitude, I'm using Windows 7. But I want to be able to develop in the evenings to keep working through the evenings under Windows 10. And Windows 10 strongly resists having anything to do with that original SMBv1. You have to do a whole bunch of things to get that working.

I also ended up investing, spending, consuming, and losing a week trying to get the Open Watcom remote debugger to work. It's the only remote debugger that can still debug a DOS target machine from Windows. It's incredibly finicky. But I did finally manage to get it working, only to discover that it does not handle, believe it or not, included code in source files. That is, if you try to step into or jump to code that's in an include, it shows you the proper line number, but it doesn't change the source file. And I said, really? And, I mean, I verified this with the maintainer of the Open Watcom project. And he said, yeah, we don't do that. I thought, okay, well, I can't use it.

So anyway, I finally ended up returning to my tried-and-true pure 16-bit tools. And I have finally achieved my target in spades. I have a highly efficient development and testing environment established and working. And using that, I've been writing code for Intel's AHCI controller spec when it's in AHCI mode. Remember that prior to this work, the work I'd been doing back in 2004, no, in 2013 is when I was working on - when I stopped working on SpinRite 6 in order to do SQRL. So it was in 2013 we still had legacy mode for the controller. And most systems still had that. It was something you were able to choose in the BIOS.

And so what we know is that the next SpinRite needs to know how to talk to AHCI controllers in their native AHCI mode. So that's the code that I'm writing now. Most modern systems will be in AHCI mode. And in fact, the latest AHCI hardware has dropped support for legacy mode altogether because it's not needed by any current OSes and

won't be possible to switch back into legacy mode. So I have support for that. That I wrote back in 2013. SpinRite will still have that. But what I'm working now on is full-on AHCI controller. And once I got this environment established four days ago, I plowed into the AHCI controller spec and began writing code.

What I discovered is that Intel's AHCI controller is an amazing piece of engineering. A week ago it seemed large, mysterious, daunting, and opaque. Today I can report that to me it now seems clear and obvious. I feel like I've obtained the mindset that the engineers who originally defined it had, and it all makes perfect sense. So I'm very excited to put it through its paces. And when I head back to my place with Lorrie this evening, I'll sit down and continue writing code. So I expect to have something for the early release testers to begin pounding on very shortly. So getting there.

Leo: There is, I should have mentioned, an xkcd comic eulogizing John Conway. It could have been your Picture of the Week.

Steve: Yup. I didn't see it until too late. It's very cool.

Leo: Yeah, let me refresh this.

Steve: It repeats.

Leo: It starts here with a person, yeah.

Steve: Yup.

Leo: And then there he goes.

Steve: Very nicely done.

Leo: Yeah. That's really sweet, actually. I think Conway would have loved it.

Steve: Yeah. Okay. So what we know is that Apple and Google have engineers, crypto people; have gotten together and designed something which they are immediately implementing. I think they're working, both of them, on putting this into APIs on their respective platforms, to which applications can be written. But they've also said that, because they recognize that the potential need for contact tracing is not going away anytime soon, they intend to, as soon as they can, submerge this technology into the underlying OS, so contact tracing with this design will be part of both iOS and Android moving forward. So what I want to do is first explain exactly what the technology is, exactly how it works. And then, Leo, you and I will discuss the implications and upsides and downsides and spoofing and all the other things.

Okay. And I should just mention that even when this is submerged into the phone, it's interesting, notice it's not contact tracking, because I'm sure "tracking" is a bad word, it's contract "tracing." But even so, I'm sure there will be a setting to turn it off if you don't want that feature on in your iOS or Android device. So all of this is explicitly opt-in and if

the user chooses to participate in this. And I was listening to you talk about this in MacBreak Weekly and agreeing that this potentially offers enough collective benefit, and I think we will see from the technology that they have really dealt with the arguments that uninformed people have that this makes a lot of sense.

Okay. So when contact tracing is first brought up on a device - iOS, Android, whatever. And you're right, Leo. For people who don't have smartphones, it would be entirely possible to have a little credit card-size puck of some sort which is only a Bluetooth beacon thing, to have like some lower cost alternative, if there was some reason to do that. So you could certainly do that. So each instance synthesizes from high entropy a single master tracing key. So that is, so it's a 256-bit master ID for the device. There's only one of them. It is well protected, never a need for it to be changed. So each device has this single master key.

From that a tracing key - so I'm sorry, there's the master tracing key. From that a daily tracing key, which is half the size, it's 128-bit, is deterministically derived from hashing the master tracing key with the Unix epoch day number, which we know is the number of days since January 1st, 1970. So this tracing key, the daily tracing key, changes every day, once per day, and it's derived from the master tracing key. But because the daily tracing key is the output of an HMAC-based key derivation function, the master tracing key is never revealed, and the daily sequence of daily tracing keys cannot be predicted. So however, as we'll see, and this is important, the device that contains the master tracing key can itself, because it alone has the master tracing key, it can recreate any previous days' daily tracing keys that it may choose to. And we'll see why that's important here in a second.

Okay. If the user of the device should subsequently test positive for COVID-19, the user can instruct the device to upload to a regional health server a block of previous days' daily tracing keys for the period during which they may have been contagious. So we'll just sort of hold that thought for a second because there's more we need to explain. But I wanted to explain where - so there's the master key. The master key is hashed with the day in order to produce a daily key which, because the day changes every day, that daily key changes every day.

But the way this is designed, the way they designed it this way is that, if at some point, any point in the future you were to test positive for COVID-19, you could say, oh, shoot, and you could upload to a shared server - and we'll talk about the way that's done in a second - the set of previous days' tracing keys. They don't reveal the master key. They're just the tracing keys that you had had on earlier days. So the point is, because they are derived from the day number, you don't need to store those daily tracing keys. You can get them any time you want.

Okay. So we've got a daily tracing key. The thing that is actually transmitted in the Bluetooth beacon is called a Rolling Proximity Identifier. Oh, and I forgot to mention because this reminds me here again, the tracing key is half size. Remember the master key is 256 bits. We want 256 bits for maximum entropy so we don't get collisions among these randomly chosen tracing keys because they're all just chosen independently. The daily tracing key is a compromise. Because they do transact online, they need to be stored in a server, and the only real need is to avoid collisions. It's a half-size key, 128-bit key, which provides still plenty of entropy. But it was chopped in half just because there was no need there for 256 bits.

Okay. So the rolling proximity identifier is also a half size, that is, chopped in half. It's a 16-byte, 128-bit value sent out as the beacon from all participating devices. We've discussed several times in the past how to prevent Bluetooth tracking. All modern devices already randomize their Bluetooth MAC addresses. Those MAC addresses change on a random schedule between once every 10 to 20 minutes, so on average every 15

minutes. So but when you see 15 minutes, think of, well, yeah, but actually it's between 10 and 20 minutes. The device chooses a time period in the future between 10 and 20 minutes to next rotate or pick a new Bluetooth MAC address.

The system's rolling proximity identifier, which is a thing that is actually sent, it is changed synchronously with the MAC address changes. So the rolling proximity identifier changes with the MAC address. When the user has enabled contact tracing, every time their Bluetooth MAC address changes, a new rolling proximity identifier is generated. That's the actual data contained by the Bluetooth packet. Since the Bluetooth MAC address and the rolling proximity identifier change synchronously, what that means is that no single identifier will straddle MAC addresses, and no single MAC address will straddle identifiers. That means it's not possible to link and track contact tracing by MAC address or vice versa.

The value of this 128-bit rolling proximity identifier is also fully deterministic, fully determined. It's derived from that daily tracing key and a 10-minute window number from the start of the day. So you take the number of seconds so far that you are into the day, divide that by the number of seconds in 10 minutes to get an integer, which is essentially the 10-minute interval that you're in. So that's something that will be changing every 10 minutes throughout the day.

So everyone who's voluntarily participating in this contact tracing system is emitting 128-bit identifiers which are ultimately derived from their master, that grand master 256-bit master tracing key, which in turn creates a daily tracing key based on the day of the epoch that we're all in together, the number of days since January 1st, 1970. Which is then used, it's hashed along with the 10-minute window of the time of day to create a proximity, this rolling proximity identifier which changes every 10 minutes. And this was cleverly done. They clearly did this on purpose. Notice that since the MAC address changes every 10 to 20 minutes, and the rolling proximity identifier is calculated on a 10-minute scale, no two sequential MAC addresses can ever have the same rolling proximity identifier. So this was carefully assembled to ensure privacy.

Okay. So we have a system where participating devices are broadcasting an identifier, which changes unpredictably from the outside once every 10 to 20 minutes. And again, because it changes with the MAC address, and the MAC address interval is deliberately fuzzed, it's on an average of every 15 minutes, and every participating device is also collecting all of the similar incoming 128-bit rolling proximity identifiers from everyone nearby. You could also use, for example, the received signal strength identifier, if you wanted to get some sense of the physical proximity of the two devices.

So, for example, the API might say, well, yes, we received the beacon, but my god, the received signal strength is so low that that's probably further away than viral contamination could occur, so we're not going to bother with that one. So there might be a receive signal strength threshold below which the beacons, even though they are received, they're ignored as just being unlikely to actually represent, the other person or you, represent a threat to either of you.

Okay. So if the device has not before seen the 128-bit rolling proximity identifier, it will be maintaining a list of them. So if it hasn't seen it, it will add it to the list. And since there's no need to retain previously received rolling proximity identifiers forever, they can be and will be deleted from the receiving devices after they've aged out at what I've seen written is 21 days. That could be changed to 28, or it could be whatever. But the point is you don't need to keep them forever.

Okay. So now, someone who's been participating tests positive for COVID-19. If they choose to, that is, presumably at their discretion - maybe they don't have a choice in China. I don't know. But here in the U.S. we probably still have a choice. At their

discretion, to help the world and to help protect their friends and family, they choose to notify everyone whom they may have been in contact with during the days prior to their diagnosis, when presumably they were contagious. So they instruct their device to notify the world. Their device alone contains that single, derived from pure entropy, master tracing key which, based upon the Unix epoch day, derives the daily tracing key. So the device goes back some number of days and uploads each day's tracing key, during which time they may have been contagious. And in the process we relabel these tracing keys "diagnosis keys." Once they have been associated with a COVID-19 positive individual, they become diagnosis keys by definition.

So think about this. All of the rolling proximity identifiers which were emitted by that user's device on any given day were derived solely from that daily tracing key and the time of day in those 10-minute windows. That means that anyone else who obtains those daily tracking, I'm sorry, those daily tracing keys, now called "diagnosis keys," anyone who obtains those diagnosis keys is also able to re-derive the same set of Bluetooth beacons that the COVID-19 positive user's device transmitted that day.

So on the receiving end, every participating user's device periodically downloads all of the daily diagnosis keys, presumably from people in their region because there's no reason to get them from countries you are not located in, and presumably only since you last downloaded them. So it might be daily thing. You might tell the server last time I checked in was this time and date. Give me any new diagnosis keys for where I am now or within some region that I haven't previously seen. For any new keys that the device has not already received, the user's device simply recreates the original rolling proximity identifiers which were originally derived from the now believed to be infected user under their tracing key. They use what we now call the "diagnosis key" to recreate the same set of rolling proximity identifiers, and they check their list.

Remember, they're storing all the ones that they've seen, so they look for a collision, that is, do any of the resynthesized rolling proximity indicators match any that were received during the last 21 days. If so, the person knows they have had a potential exposure to somebody who has since claimed that they are COVID-19 positive. They have no idea who; they have no idea when. Well, actually they would know what day it was, and presumably their device might choose to tell them this was the day that a collision was found. So if there are any matches, the recipient user knows that someone whose device theirs was close enough to, presumably, well, to exchange Bluetooth beacons and presumably with a high enough received strength indication, has posted a diagnosis key that generated a rolling indicator that they received that day.

So that's the system. It is carefully thought out. I've gone through the spec. I've looked at the crypto. These guys didn't miss anything. And so that's the system that, apparently without any modification, iOS and Android will both have in short order. We need applications, and we also need to do something about the things that can go wrong.

So what can go wrong? Moxie Marlinspike, who has earned and deserved everyone's respect as a crypto-savvy researcher - he developed Signal and did a stunning job with Signal. Apparently, in what I presume are an initial bunch of tweets, he got some things wrong. And I presume it's just because he didn't have time to sit down and read through the spec carefully because, for example, he was immediately worried about MAC address tracking, and the spec explicitly explains how that cannot happen. There is no fixed Bluetooth MAC address that can be associated with these tokens by design.

He also talked about huge amounts of bandwidth being consumed. I assume that he was presuming that the tokens themselves are what would be transacted, rather than the diagnosis keys. One of the cleverest things about this is that we have HMAC-based keying that only allows these things to be generated in a forward-going manner; and the diagnosis key, knowing the day, allows you to resynthesize all of the individual rolling

keys. So anyway, these guys, from a crypto standpoint, they have produced a very powerful system.

One non-malicious problem that could arise is that radio proximity does not perfectly match viral propagation proximity. In other words, the way I think of it is your device, for example, might be exchanging beacons with people in the apartment above, below, and to either side of yours. But depending upon how well this received strength indication is used, assuming it is, that doesn't mean because you can receive their radio signal that they would represent a source of viral contagion. So the system could potentially false positive if you happened to have your phones on either side of a wall so that the signal strengths were strong, and somebody on the other side of the wall later came down with COVID-19.

Leo: What's the range of LE? It's less than regular Bluetooth, I would bet.

Steve: That's a really good question. I don't know. You're right. I don't know.

Leo: It's Low Energy, so that sounds like it must be.

Steve: Yeah. We ought to - that's a really good point. I was just assuming it was the same 10 meters, thus 30 feet. But it may be that it is enough shorter that, like, you're already going to need to be within sneezing distance in order to reach the other person. And of course the other glitch is, and this is something I haven't seen, I haven't read anything about, is you don't want people spoofing their COVID-19 infections onto the server. So, for example, you wouldn't want somebody going to a concert and spending the night in proximity to as many people as they possibly could and then misrepresenting the fact that they're now COVID-19 positive and causing a false positive alert to everyone.

So we do, I mean, if nothing else, the Zoom fiasco, the Zoom startup fiasco has shown that the social engineering side also needs to have attention paid to it. We've nailed the technology side. So I don't know how we'll handle that. It needs to be done correctly. But we really do have a beautiful technical solution, I think. There's no notion of user identity. It's not encoded. It's not encrypted. It just isn't there. These things are coming out of HMAC, and it's just random gibberish. There's no sense of location, that is, the protocol...

Leo: So it's not related to - you can't derive the randomly generated MAC addresses from this HMAC. That was one of Moxie Marlinspike's concerns was all of a sudden somebody tests positive, he's going to have - it's going to publish all of his rotating Bluetooth MAC addresses to the world.

Steve: So there's two different things here.

Leo: I know there's these derived ones.

Steve: Well, so there's the MAC address, and that's separate from the payload of the Bluetooth packet, the rolling proximity identifier. That's the data contained by the Bluetooth packet.

Leo: But that has to be able to link back to the person who generated it at some point, if you test positive.

Steve: Yes, correct.

Leo: So you must be able to derive the MAC address from that.

Steve: Not the MAC address. Again, you need to - the MAC address is different from the rolling proximity identifier. The MAC address is the communications link. The rolling proximity identifier is the 128-bit tag carried in the payload.

Leo: And that's generated randomly.

Steve: Well, that's generated through this HMAC, which is clever because it allows them to be recreated from the daily tracing key, which is then called the diagnosis key.

Leo: But it's a one-way transmission.

Steve: Yeah, it is a one-way function. But again, you put the same thing in, you're going to get the same thing out, which is the elegance of this, because it means you just post your diagnosis keys, the keys for the days that you were contagious, and everybody who receives those can then regenerate the whole day's worth of rolling proximity identifiers to see if they have any collisions, if they have any matches.

Leo: So he says the moment you test positive - he misunderstood it. He says all of your BTLE MAC addresses over the previous period become linkable.

Steve: Yes, that is not true.

Leo: That is not the case, okay.

Steve: Yep, not the case.

Leo: That would certainly be an issue, obviously, because the reason you rotate this - because it's a privacy issue.

Steve: Yup, and it's also very clear that Apple and Google knew this because they deliberately made the 10-minute period on the inside of when the MAC address rotates or changes. So there is just absolutely - oh, and they also made them change synchronously, and the spec specifically says that. So I just think that Moxie, you know, maybe he, I don't know, had a bad hair day. You've seen him, so that's not much of a stretch. Who knows? But yeah, there is no MAC address linkability. They really nailed the technology. It's beautiful.

Leo: Right, okay.

Steve: So we need to solve the social engineering problem. We need not to allow people to post to what will what become an important tracing database if they aren't actually recently shown to be positive. But, you know, that could be under the control of local health officials. And we don't yet know how they're going to be distributing or managing antibody testing. But the cool thing is you can't have tracing unless you have the technical foundation. To their credit, Apple and Google instantly stepped up and created, as far as I can see, a perfect technology base on which can be built the tracing that we probably need throughout the world moving forward.

Leo: Yeah, it's clever. They say that Apple and Google will push out these APIs in the next couple of weeks, or mid-May, I guess, so it's about a month from now. And so they'll be available to apps at that point with operating system updates. And then apps will come later from those two. There'll be reference apps both from Apple and Google.

Steve: Oh, I mean, I could write this overnight. I mean, it's really simple. And that's what you want. You want a clean, simple, obvious, obviously provably correct system. And they nailed it.

Leo: That's the best place to start, yeah.

Steve: Yeah.

Leo: Well, I'm relieved. That seems good.

Steve: And there we have a podcast.

Leo: Very interesting.

Steve: And all of our listeners know how it works now. And I have a feeling it may be in our lives in the future, yeah.

Leo: It's very elegant, yeah, it's very elegant. Yeah, I think it's very interesting.

Steve Gibson's at GRC.com. That's where the podcast lives, as well. GRC is the home for SpinRite. That's that thing he's working on, SpinRite 6.1. You'll find the world's best hard drive maintenance and recovery utility at GRC.com, along with a lot of other freebies including ShieldsUP! and of course this show - 16Kb audio, 64Kb audio, and very nicely done transcriptions, all available at GRC.com.

We have 64Kb audio and video, as well, at our website, TWiT.tv/sn. It's also on YouTube. You could subscribe to the Security Now! channel there. And if you will, subscribe to the podcast. That way you'll have every episode. You won't have to think about it. Just pick a podcast appliance and subscribe. You can even listen on

Amazon's Echo - that's not a subscription, but you can listen - or the Google Home just by saying, you know, "Hey, device, play Security Now! podcast," and it will.

Steve, we'll be back here next Tuesday around 1:30 Pacific, 4:30 Eastern, 20:30 UTC for another thrilling, gripping, sassy edition of Security Now!.

Steve: Absolutely. Today was Patch Tuesday. Not enough time to deal with all that. We'll figure out...

Leo: Was there a patch?

Steve: Oh, yeah, I got my Win10 system is, like, panting for me to shut it down so it can update. So, like, okay. Hold your horses.

Leo: Talk about it next week.

Steve: Gonna happen right now. Talk to you then, my friend. Bye.