Zoom's E2EE Design

Description: This week we look at which browsers still permit drive-by website downloads, Google's plan to blacklist notification-abusing websites, a deeper dive into local PC port scanning being performed by websites, Facebook's move to tighten up on high-impact posters, the new lawsuit against Clearview AI, some very interesting strings found embedded in Google's latest messaging app, the very worrisome return of a much more potent StrandHogg for Android, the refusal of SHA-1 to die, a more powerful new USB fuzzer, and an update in some nearly finished SpinRite work. Then we take a look at Zoom's newly detailed plans to become the world's most secure teleconferencing platform.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-769.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-769-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Coming up on this episode, the ACLU sues face recognition company Clearview AI. We'll talk about end-to-end encryption, both in Google's new RCS messaging system and in Zoom. How do they do it? Steve's impressed. Stay tuned for the details.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 769, recorded Tuesday, June 2nd, 2020: Zoom's E2E Encrypted Design.

It's time for Security Now!, the show where we cover your security online, your privacy, how things work, and a few little tidbits thrown in here or three with this guy right here, Steve Gibson of the Gibson Research Corporation.

Steve Gibson: How things work and what happens when they break.

Leo: When they don't work.

Steve: And stop working, yes.

Leo: Salute to you, Steve. Hello.

Steve: Leo, great to be with you once again, Episode 769 for this first podcast of June. God, where is the year going? 2020, what a bizarre year we're in. Friday before last a

large team of folks working on behalf of, in some cases with and some cases helping Zoom, published their end-to-end encryption design document, 25 pages of aspiration and clearly some perspiration. Lots of equations that we're not going to try to do on a mostly audio podcast. But the concepts are there, and they're readily understandable. So I thought it would be fun to take a look at how Zoom is - what they've already done and at what their plan looks like, the roadmap that they've laid out for how they're going to get to what will arguably be the most secure teleconferencing solution on Earth by the time they're done.

So today's topic is Zoom's E2EE, which of course is the acronym we're now seeing more and more, end-to-end encryption design. But lots of fun stuff happened this week. Well, fun if you're in the podcasting business. We're going to look at which browsers still permit drive-by website downloads and define what that is. Google's plan to blacklist, I mean, bad list notification-abusing websites. We're going to take a bit deeper dive, thanks to some research that Lawrence Abrams did over at BleepingComputer into all the other sites that are apparently doing local PC port scanning. We talked about eBay last week. Turns out, yes, there's 300 and some odd others. Many of them are substantial.

There's also a new lawsuit being brought against Clearview AI, which I think is nothing but good because we need the brakes to be put on this, and some serious thought to be given to the consequences of this pervasive facial hoovering of all images everywhere and then AI to identify them. There's some new facts that come out in this lawsuit that I want to share with our listeners. Some very interesting strings have been found embedded deep in Google's latest messaging app, indicating some future intent, which is intriguing. We've got a very worrisome return of a much more potent version of StrandHogg for Android. We also had the refusal of SHA-1 to die, despite many attempts to just end it. It's just really putting up a fight. A more powerful new USB fuzzer and what it has found.

I will update our listeners on some very nearly finished SpinRite work, sort of a necessary subproject that I've been on for about three weeks, pretty much wrapped up yesterday with a little bit more fine-tuning to do. And then, as promised, we'll take a look at Zoom's new detailed plans to become the world's most secure teleconferencing platform. And this Picture of the Week is not apropos of anything security. But I encountered it when I was looking to see whether last week's wacky Windows 10 airplane was available as wallpaper. And this was somewhere nearby. And I thought, oh, this is - at least it's technology. So it's just too fun. So I think another great podcast for our listeners.

Leo: It's sort of technology. Sort of technology.

Steve: So Leo, remember that picture we had a couple years ago, there's a plastic pail half filled with dirt, and there was a rod stuck into it and a big green ground wire wrapped around the rod.

Leo: Yeah, yeah.

Steve: And some person figured, oh, well, I was told I had to ground this.

Leo: There you go. Yeah, it's in the ground.

Steve: So I stuck it into a pail of dirt. And it's in the ground; right? I mean, it's dirt. So this picture sort of reminds me of that. In a perfect world, you'd have this pipe going along that needs to apparently go into a pipe that goes into the ground, and it would just do a right angle and go from horizontal to vertical into the ground. But in this world, which is far from perfect, there's a big rock in the way.

Leo: Apparently too heavy for the plumber to move.

Steve: And I do not know, I would love to know the true back story here, how this happened. But maybe the rock is not movable. It's cemented in place. Or as you said, it's just too heavy. I mean, it's a big rock. But anyway, whoever had the chore of connecting the horizontally running pipe to the pipe coming out of the ground figured, well, there's a rock in the way. So very much reminiscent of that Three Stooges episode where Moe or Curly or whoever it was was in the bathtub, trying to plumb himself out, and ended up in a pipe cage, this thing does a right turn to go up, then another 90-degree more leisurely bend to go horizontal across the top of the rock, then sort of a 45-degree angle down to get...

Leo: I like that. I like that choice.

Steve: ...to the other side of the rock, apparently because the...

Leo: It's really contoured to follow the rock's surface.

Steve: It really does. And then another 45 to get vertical in order to go into the pipe in the ground. So again, you know, I don't know, like, what the pipe in the ground is about, where that goes or where it's coming from. It must be the source of water.

Leo: I just hope this isn't a drain.

Steve: And so anyway, as I said, last week I was looking for the more sources for the wacky Windows 10 "We finally fixed all the problems" that had the landing gear coming out of the roof of the cockpit and the wings on sideways. And I thought, okay, this is just too fun for the podcast.

Okay. But so what's not fun is the surprising state in 2020 of drive-by malvertising downloads. Yesterday, Monday the 1st, a guy named Eliya Stein, who's with the advertising security firm Confiant, C-O-N-F-I-A-N-T, blogged about some research he'd just finished regarding the susceptibility of our current web browsers to so-called drive-by downloading. It's interesting that downloading can be automated, which is something you may not have thought about before, but I encounter this at some funky sites.

SourceForge is one that comes to mind. When you want to download from SourceForge something, you say, you know, I want to go to the download page. And you get, rather than like a dialog that pops up that says, hi, want to download this from here? Instead, you kind of get this little green "Be patient" spinner that says your download will begin shortly. And then without you doing anything, you'll notice if you're using Chrome, for example, that down in that lower left area of the browser window, where downloads appear, it'll suddenly open up all by itself and start downloading. And you didn't do

anything. I mean, you didn't press any buttons. You didn't get a confirmation dialog. It just did it for you.

So that's good if you're at a site where you deliberately went to a page and you want to download something. Not so good, you can imagine, if you're on a site who's hosting a malicious ad in an iframe, and this can all happen within an iframe. So in a drive-by download, iframed script content which is coming from a third-party domain, it's in an iframe to offer some containment, and the iframe points to an ad server that then provides the content on a rotating ad basis. So that's the way any contemporary ad server works. In this case, it's running in an iframe.

But thanks to the fact that JavaScript is able to directly modify the DOM, that's the web page's Document Object Model, that is, I'm sure that someone somewhere thought, oh, wouldn't it be cool if script could write its own web pages? It's like, yes, what could possibly go wrong? So in this case malicious JavaScript is able to programmatically define a new download link, an "a" tag, an <a href> in HTML parlance, then click it in order to initiate its own, from the user's standpoint, zero-click download.

Eliya's attention was drawn to this practice when BoingBoing was delivering for some period of time malicious downloads to their visitors using this technology. And after many complaints, BoingBoing disclosed that their site had been hacked. So in this case the malicious code was placed onto their servers so that it was running in the same origin as the page and had unfettered access to the user's page, very much like if SourceForge was hacked and I went to that download page. I don't know. I have no control over what's happening. It just stuck something in my Download directory.

So Eliya wrote that: "Over the following weeks, we detected this attack on a multitude of sites. Usually this manifests through a CMS [Content Management System] compromise that introduces this malicious payload. To date we've identified the presence of the IOCs [Indications of Compromise] associated with this attacker on 15 sites within our telemetry. But a cursory look at VirusTotal shows that activity around this particular payload has been ongoing for at least a year, and that these command-and-control domains are leveraged by multiple variants of the malware."

So having uncovered the mechanism that this attacker uses to drop the malware, Eliya decided to conduct an audit of recent popular browser versions to see how they handle downloads that are not initiated by direct user interaction. The inspiration for doing this analysis was the surprising discovery that most browsers will honor downloads triggered from cross-origin iframes, which is, like, shocking. Like why would you ever want that to happen, especially with no user interaction? And what's more, these zero-click downloads are often still possible within sandboxed cross-origin iframes. Remember that sandboxing is an extra parameter that can be added to an iframe's attribute list to, like, further strengthen its controls.

In one example, this has only finally been addressed in Chrome for this latest Chrome 83 that we all got and talked about last week. And I have a link in the show notes to chromestatus.com/feature and then a long serial number-ish thing. It reads - so this is for Chrome. This is in the Chrome status log, download in sandboxed iframe (removed) under the security subhead. And it says: "Sandboxed iframe can initiate or instantiate downloads." And it's like, what? Really?

Leo: Doesn't seem like a good idea.

Steve: Like before last it could? Unbelievable. So they said: "Chrome is planning on removing this capability." Yes, please. And they said, i.e., "Chrome is going to block all

downloads initiated from or instantiated in a sandboxed iframe by default." What a concept. "The embedder may add 'allow-downloads' to the sandbox attributes list to opt in." Please don't. "This allows content providers to restrict malicious or abusive downloads." Well, wouldn't that be nice. Then they said: "Removal is expected in Chrome 83. This allows content providers to restrict malicious or abusive downloads." Yeah, good.

So anyway, finishing this up under Comments, it says: "Downloads can bring security vulnerabilities to a system." Who's writing this? Anyway: "Even though additional security checks are done in Chrome and the operating system, we feel blocking downloads in sandboxed" - hello, sandboxed - "iframes also fits the general thought behind the sandbox." Oh, really. Yeah, okay.

Anyway, so Eliya wrote: "For our study, we will test the mechanism inside and outside of cross-origin iframes. The payload we use looks like this." And I've got the short little bit of JavaScript code, just because it was so cute and short and understandable. So basically there's a document.createElement, and then in quotes "a." So it's going to create an "a" element, an "a" tag, you know, an anchor. And that's being assigned to the variable link. Then link.setAttribute, href, payload.apk. So that in this nascent "a" tag, the href value is set to, or the href tag, the href name is set to the value payload.apk. So that's what will be downloaded when the link is clicked.

Then the attribute "download" is set to download, to indicate what kind of link this is. Then the display style is set to "none" because of course, right, you don't want the person, the user, the hapless unwitting victim who's gone to this web page to see any of this going on. So then to the document body the link is appended. And then, in the final line of this little JavaScript code, the link is clicked. Which of course initiates the download of payload.apk. What could possibly go wrong?

This simple bit of code does what I described before: It creates a new link where there was none, sets its download href, makes it invisible, appends it to the existing web page, and then programmatically clicks the link to initiate a download.

So what did Eliya learn about the status of today's web browsers regarding their hosting of these ad hoc download links in iframes? Chrome 83 does indeed block downloads initiated from sandboxed, thank goodness, cross-origin iframes. But apparently Chrome 82 didn't. Well, fortunately we all have 83 now. But even it, even Chrome 83, is still willing to drop a file if the iframe's sandbox parameter is not set.

So let me say that again. If on all your pages, anywhere you're using an iframe, if you haven't gone back and added "sandbox" to the iframe, it's not sandboxed. And malicious ads can load some script that will download a file. Unbelievable. But true. However, other desktop browsers will trigger a prompt for the download. This must have been some ease-of-use thing that Google thought Chrome should have, where just kind of like, oh, look, it's downloading something. I hope that's good. Oh, my goodness.

Anyway, other desktop browsers will trigger a prompt for the download, including the security-conscious Brave and Firefox. Both of them give you like an OS dialog. And under many circumstances, browsers will drop the download without any prompt at all. For example, the same payload in a non-sandboxed cross-origin iframe in Chrome 83 will drop the download without a prompt and without any indication that the initiator is not the origin that's displayed in a URL bar. In other words, it's not same-origin, it's some other origin. It's Ads R Us and TrustUs.com.

And at least here Firefox's prompting for action is consistent, that is, Firefox always prompts. You never get an unattended, unprompted download. And apparently Safari attempts to honor and initiate the download, but then kind of gets stuck in some way that he wasn't clear about along the way. And in the mobile world, behavior is also

inconsistent. Android browsers are, fortunately, quick to warn when the download is a file with an APK extension. So they're all tuned-up on that. But any other type of file often doesn't receive a prompt. And as we know, files could be renamed after they have arrived. So not clear how much protection that adds.

At this point in his blog posting Eliya places a large bold callout reading: "It's 2020, and we can still force downloads that are not user initiated, without any prompt, from cross-origin iframes, in half the major browsers out there. Why?" Incredible.

He concludes by observing: "Within the context of hacked sites as perpetrated by our site-compromising attacker, ad tech has no role to play." Meaning that they are, because they're same-origins events because the site delivering the page is the one that was compromised, as opposed to an ad server running in a cross-origin iframe. He says: "No amount of ad or iframe sandboxing can help to mitigate the last mile of this particular attack because the download is dropped from a seemingly trusted source. So perhaps we should think about always prompting users before a download takes place, along with an informational modal dialog about the origin of the download."

So there he's saying, look, because you can have - you can't rely on malicious content always coming cross-origin. Even if the browsers were all handling that correctly, which they're not, it's certainly the case that sites are getting compromised. And by using this download automation for apparently ease of use and user friendliness, maybe all downloads should be user prompted. And I completely agree. There's a point where security really ought to trump automation convenience. Call me old-fashioned. I'm sure I am. But I would like to be in the loop anytime my browser is downloading a file into my machine. I just can't imagine that you can secure things otherwise. So thank you, Eliya, for that research and for a bit of a heads-up on this.

Also in Google news, they're planning to blacklist - and as I wrote that I thought, whoops, I mean bad list...

Leo: Thank you. This would be a bad day to get that wrong.

Steve: Yes, notification abusing sites. They intend, Google intends to shield Chrome users from two types of abusive alerts. And remember we were just talking about alerts a couple weeks ago when I saw one that just made my eyes cross. And Leo, you said, wait a minute, you mean you have to acknowledge this notification to allow notifications in order for the site to allow you in. It's like, yeah, that actually happens.

Leo: Yeah, that pisses me off. That's a site I'll never visit, yeah.

Steve: And this is what Google's talking about. They call them "permission request issues" which attempt to mislead, trick, or force users into allowing notifications. And of course this is the behavior I talked about a couple weeks ago. The second form of abuse are the content of subsequent notifications which are then used for phishing, delivering malware, or faking chat messages, warnings, or system dialogs. So with the forthcoming mid-July release of the next Chrome, Chrome 84, the browser will also inform users that these abusive websites may be trying to trick them by phishing for private information or promoting malware.

A Google spokesperson explained that: "Abusive notification prompts are one of the top user complaints we now receive about Chrome." And I have to say I'm just not clear on the whole value of the notification. Remember that, if you allow a site to enable

notifications, then even when you're not at that site, even when your browser is not forefront, the notification system now links into the OS notifications, and you see notifications from that site that look undistinguishable from OS-grade notifications. I mean, who thought that was a good idea? Anyway...

Leo: People who are pushing browsers as applications.

Steve: Yeah, exactly. Although Chrome 84's abusive notification protection is designed to only trigger for new notification permission requests from abusive websites, Google is also planning to expand these protections to users who've already provided abusive sites with permission to deliver notifications. Yay. Because we've talked about what happens if you've already given permission. We talked about how you can go back and audit your notification list. It's not hard to do. But lots of less-focused users don't know how to do that. Google said only a small fraction of websites will be affected by this change. But we expect the impact on notification volume will be significant for some users.

So what's happening is there's only a small number of abusive sites, but they're being very abusive. So what they're going to do is they're going to get proactive. They plan to give these sites that have been identified as abusive 30 days' written notice by email prior to tagging them as abusive. And that'll allow websites' owners to check whether they've been tagged using the Google search console in order to check the status of their site. They'll have 30 days to fix their content and ask to be resurveyed in order to be taken off of the pending abuse list. But starting in August they will be actively blocking sites, these sites that have been tagged as abusive. So essentially they're going to proactively step in to say, you know, this is not how we intended site notification to work. Cease and desist or you're in trouble.

Leo: Yes?

Steve: I just lost my picture.

Leo: You paused?

Steve: Ah. Interesting. Anyway...

Leo: You don't need pictures. You never wanted video in the first place.

Steve: It just went weird. This is Skype these days. Ah, thank you. Anyway, so starting in August, in addition to that, Chrome will also be blocking resource-consuming ads which tie up too much of the system's resources without the user's knowledge. So bravo to Google. These all seem like clean, sensible, and responsible actions from the industry's number one browser source.

So, as I mentioned at the top, the question of who else might be doing this eBay-like ThreatMetrix port scanning that we talked about last week. Lawrence Abrams over at BleepingComputer was wondering, as he frequently does, and decided to find out. To determine which other sites might be using the same scanning script, BleepingComputer reached out to a company called DomainTools that's a cybersecurity company specializing in web domain and DNS threat intelligence. So they're the right kind of

profile. It turned out that when websites load ThreatMetrix's antifraud script, they load the script from a customer-named online-metrix.net hostname.

Okay. So, for example, eBay loads the ThreatMetrix script from a domain src.ebay-us.com. Meaning that the ThreatMetrix script appears to be coming from eBay. But a query to that src.ebay-us.com domain returns a CNAME record. That's CNAME for canonical, and it's also known as an alias. In other words, it's the src.ebay-us.com domain is an alias for another domain. In this case, that redirects DNS lookup to h-ebay.online-metrix.net. That is, over to the actual ThreatMetrix domain.

So as a consequence of that, that's all visible. So DomainTools was able to provide BleepingComputer with a list of 387 unique online-metrix.net references that share a similar naming scheme. And Leo, I've got a link down toward the end of this page to a Google spreadsheet list of these names. It's rather daunting when you actually see it. So then, using this list, Bleeping Computer visited the sites for many of the larger prominent companies to verify whether they were in fact performing a local client-side, that is, browser-hosted port scan, into their visitor's computer when someone just visited.

BleepingComputer wrote that they did indeed detect port scanning when visiting Citibank, TD Bank, Ameriprise, Chick-fil-A, LendUp, Beachbody, Equifax IQ Connect, TIAA-CREF, Sky, GumTree, and WePay. Lawrence noted that the specific scanning behavior tended to vary from site to site. Citibank, Ameriprise, TIAA-CREF immediately port scanned their visitors' computers when visiting the main page of the site. TD Bank, Chick-fil-A, LendUp, Equifax IQ Connect, Sky, GumTree, and WePay only port scanned when visitors attempted to log in. And Beachbody.com only port scanned when checking out.

This detection success suggests that many of the other well-known sites on the list are also using ThreatMetrix and likely scan their visitors at other sensitive times. There are other major names: Netflix, Target, Walmart, ESPN, Lloyd Bank, HSN, Telecharge, Ticketmaster, TripAdvisor, Paysafecard, and possibly even Microsoft. For those sites, BleepingComputer was unable to trigger a scan. But it seems likely that the scans may be present on site pages that weren't visited during their testing. Lawrence posted a Google sheet, as you had up there on the screen a second ago. I've got a link in the show notes for anyone who's interested. I mean, it's extensive.

So he concluded his report by noting that these port scans can be blocked using uBlock Origin in Firefox. Unfortunately, uBlock was unable to block the port scans in the new Microsoft Edge or Google Chrome because extensions can no longer have adequate permissions to uncloak the DNS CNAME records. And we talked about this at the time. This is something that the Chromium browsers did in the name of security, limiting the abuse of extensions.

On the other hand, sometimes you want to trust an extension to be able to do good things. And this lack of access is something that uBlock Origin's quite curmudgeonly author Gorhill mentioned a while back. He was not happy that he was no longer able to do as much as he wanted to. But Bleeping Computer also tested the Brave browser, and the port scans were allowed through. So Firefox plus uBlock is the only effective blocker.

Leo: Yay.

Steve: Of these local - yeah, I know.

Leo: That's what I use, yeah.

Steve: That happens to be where I am. In fact, something funny happened last night that we will be getting to in a minute.

Leo: It is conceivable these are legitimate scans. I mean, this is, as we had talked about last week, a threat scan; right?

Steve: Yeah. Apparently, based on Lawrence's feedback from his article last week, many users, you know, I don't think it's a skewed demographic.

Leo: I don't want it, understand.

Steve: It's like, I don't want my browser scanning my own computer.

Leo: Right.

Steve: Hands off. But that's being done.

Leo: You have to trust the source. You know, I see American Express is on here, for instance. That's a financial institution. I think they're probably doing it for a good reason; right?

Steve: Yeah.

Leo: But you have to trust them, like that they're not going to look at what else I've got going on, you know, that kind of thing. So, yeah.

Steve: Well, so Facebook has decided to require some identity verification for high-impact posters.

Leo: So I have gotten calls on the radio show. They're saying, hey, Facebook wanted me to send them my driver's license. They're doing that.

Steve: I guess they should be flattered because it means that they have some reach.

Leo: Yeah, they're important.

Steve: In Facebook parlance. Last Thursday the Facebook blog was posted verifying the identity of people behind high-reach profiles. This is really short, so I'll just share it verbatim. Facebook said: "We want to ensure the content you see on Facebook is authentic and comes from real people, not bots or others trying to conceal their identity. In 2018, we started to verify the identity of people managing Pages with large audiences, and now we're extending ID verification to some profiles with large audiences in the U.S.

Moving forward, we'll verify the identity of people who have a pattern of inauthentic behavior to Facebook and whose posts start to rapidly go viral in the U.S. We want people to feel confident that they understand who's behind the content they're seeing on Facebook, and this is particularly important when it comes to content that's reaching a lot of people.

"If someone chooses not to verify their identity, or the ID provided does not match the linked Facebook account, the distribution of their viral post will remain reduced so fewer people will see it. In addition, if the person posting is a Page admin, they'll need to complete Page Publishing Authorization and will not be able to post from their Page until their account is verified through our existing Page Publisher Authorization process. IDs will be stored securely and won't be shared on the person's profile." That's, I think, a key that I'll come back to in a second.

Leo: Yeah. That's what people are worried about, yeah, exactly.

Steve: Yes, yes. They said: "Visit the Help Center for more information on the types of IDs we accept." And they said: "This verification process is part of our ongoing efforts to create greater accountability on Facebook and improve people's experiences on our apps."

So this is another - we have a couple more little bits. Actually the next thing I want to talk about is Clearview that demonstrates we've got problems to solve. But it's clear that, as a global society, we're currently struggling to figure out how to manage this new Internet-enabled social media. We know anonymity is a powerful tool for speech freedom, but that freedom can so easily be abused that some privacy-protected accountability seems like it's going to be a necessary step as we learn to manage what we have created here.

During this past weekend's energetic protest events, many of the authentic protestors were only too happy to share their identities. They were there for what they felt was a righteous cause. But none of those who were looting the stores wanted to be known. Everybody was in a big anonymous group, and a bunch of people were taking advantage of that anonymity. And as I said, Facebook's posting notes that IDs will be stored securely and won't be shared on the person's profile. So again, I think as long as they make that clear, and if they're presumably able to honor that, not let that identity information leak, then that seems like a useful thing. It will probably go a long way toward beginning to get this under control.

Leo: I guess the problem is that people just don't trust Facebook with their personal information; you know? They're worried. I don't know what I would do if it asked for that. Fortunately, I'm not viral enough.

Steve: Well, they're certainly using it as a conduit to get access to mass amounts of people, massive reach. So I think it seems like to me a useful tradeoff.

Leo: There's a price to pay, a fair price. Yeah, yeah, I agree.

Steve: I don't know how else you control this.

Leo: Right.

Steve: I mean, literally, it's otherwise completely uncontrolled.

Leo: Yes, troll farms.

Steve: So Clearview AI is in the crosshairs once again. This time it's the ACLU that's decided that Clearview AI needs to be stopped. The headline of the American Civil Liberties Union website is full of righteous indignation, with the headline: "We are taking Clearview AI to court to end its privacy-destroying face surveillance activities." Then the subhead is: "The company's surveillance activities are a threat to privacy, safety, and security." But I need to stop here for just a moment to note the oh-so-delicious irony of the fact that upon visiting www.aclu.org last night, my trusted Firefox browser popped up a warning which reads: "Firefox blocked a fingerprinter on this site." I'm not kidding, Leo. I had the presence of mind to snap the screen, and it's in the show notes here.

Leo: Oh, yeah, yeah. Oh, I see this all the time, yeah.

Steve: Yeah. So that's just too wonderful. The American Civil Liberties Union on their website, which is about to get all wound up over the "privacy-destroying surveillance activities" of the Clearview AI company, is using web browser fingerprinting to track their site's visitors.

Leo: Do they tell you what fingerprinter or what technology it's using on that?

Steve: I didn't go any further. So we don't know. And I don't have any back story on this. You know, the ACLU is...

Leo: They don't have ads on that site, so...

Steve: No. And they are a highly charged, often controversial organization. So perhaps they have some security or safety-related reason for adding non-browser cookie tracking to their site. But I thought it was funny to receive this warning when going to a pro-privacy, anti-surveillance site.

Leo: Yeah, that's weird, yeah.

Steve: Like the ACLU. It's like, yes, but we're going to try to fingerprint you. Anyway, but that aside, I agree that Clearview's arbitrary photo facial surveillance and image recognition capability represents another aspect of the Internet we have unleashed. When you combine an Internet-connected camera in everyone's pocket with real-time fixed camera surveillance, massive cloud storage, and cloud computing resources, you get consequences that were unintended. Like you take a whole bunch of innocent things; but when you aggregate them, you can end up with something that's not so innocent in aggregate.

So what the ACLU wrote and posted last Thursday does contain some new information, a couple things where I said, "What?" So I want to share it. They said, and a lot of this we know, but this sort of sets up for this new stuff: "For several years, a little-known start-up based in New York has been amassing a database of billions of our faceprints" - which is the term they're using here - "unique biometric identifiers akin to a fingerprint or DNA profile, drawn from personal photos on our social media accounts and elsewhere online. The company has captured these faceprints in secret, without our knowledge, much less our consent, using everything from casual selfies to photos of birthday parties, college graduations, weddings, and so much more.

"Unbeknownst to the public, this company has offered up this massive faceprint database to private companies, police, federal agencies, and wealthy individuals, allowing them to secretly track and target whomever they wish using facial recognition technology. That company is Clearview AI, and it will end privacy as we know it if it isn't stopped. We're taking the company to court in Illinois today on behalf of organizations that represent survivors of sexual assault and domestic violence, undocumented immigrants, and other vulnerable communities. As the groups make clear, Clearview's face surveillance activities violate the Illinois Biometric Information Privacy Act (BIPA)" - which of course we've discussed extensively here in the past - "and represent an unprecedented threat to our security and safety.

"Face recognition technology offers a surveillance capability unlike any other technology in the past. It makes it dangerously easy to identify and track us at protests, AA meetings, counseling sessions, political rallies, religious gatherings, and more. For our clients organizations that serve survivors of domestic violence and sexual assault, undocumented immigrants, and people of color this surveillance system is dangerous and even life-threatening. It empowers abusive ex-partners and serial harassers; exploitative companies; and ICE agents to track and target domestic violence and sexual assault survivors, undocumented immigrants, and other vulnerable communities.

"By building a mass database of billions of faceprints without our knowledge or consent, Clearview has created the nightmare scenario that we've long feared, and has crossed the ethical bounds that many companies have refused to even attempt. Neither the United States government nor any American company is known to have ever compiled such a massive trove of biometrics." And there I would say the operative word is "known to have."

"Adding fuel to the fire" - get this - "Clearview sells access to a smartphone app that allows its customers, and even those using the app on a trial basis, to upload a photo of an unknown person and instantaneously receive a set of matching photos." And we know that behind those photos is a rather extensive set of information that Clearview has amassed from the source of all these photos. So you get names and addresses and other available information.

They said: "Clearview's actions clearly violate BIPA. The law requires companies that collect, capture, or obtain an Illinois resident's biometric identifier such as a fingerprint, faceprint, or iris scan to first notify that individual and obtain their written consent. Clearview's practices are exactly the threat to privacy that the legislature intended to address, and demonstrate why states across the country should adopt legal protections like the ones in Illinois.

"In press statements, Clearview has tried to claim its actions are somehow protected by the First Amendment. Clearview is as free to look at online photos as anyone with an Internet connection. But what it can't do is capture our faceprints uniquely identifying biometrics from those photos without consent. That's not speech. It's conduct that the state of Illinois has a strong interest in regulating in order to protect its residents against abuse.

"If allowed, Clearview will destroy our rights to anonymity and privacy, and the safety and security that both bring. People can change their names and addresses to shield their whereabouts and identities from individuals who seek to harm them, but they can't change their faces. That's why we're teaming up with lawyers at the ACLU of Illinois and the law firm of Edelson PC, a nationally recognized leader in consumer privacy litigation, to put a stop to Clearview's egregious violations of privacy. We're asking an Illinois state court to order the company to delete faceprints gathered from Illinois residents without consent, and to stop capturing new faceprints unless it complies with the Illinois law.

"There is a groundswell of opposition to face surveillance technology, and this litigation is the latest chapter in an intensifying fight to protect our privacy rights against the dangers of this menacing technology. Across the nation, the ACLU has been advocating for bans on police use of face recognition technology, leading to strong laws in places like Oakland, San Francisco, and Berkeley, California, and Springfield and Cambridge, Massachusetts, as well as a statewide prohibition on the use of the technology on police body cams in California. We won't let companies like Clearview trample on our right to privacy."

So they, like others who have sued previously, are using Illinois' BIPA law, since as we know it makes the strongest case. And any ruling that might arise from this will likely be challenged and appealed and may finally wind up in front of the U.S. Supreme Court. And as with the encryption question, that's probably for the best since someone, and it's the responsibility ultimately of our Supreme Court, needs to create some laws to guide us forward. I just hope they are good laws because this is still the early days. We're not at the beginning of the end or the end of the beginning. I believe we're still at the beginning of the beginning.

And there's a lot more to happen as technology continues to progress and enable things that just weren't feasible before. We've got insanely inexpensive mass storage, incredibly inexpensive and ever more potent AI to do image comparison. And of course everything is glued together with cloud stuff. So we need to decide what we as a society want to have done with all of this capability and what not.

Google Messaging appears to be heading toward end-to-end encryption, which I think is, it occurred to me, an interesting stance to take against all of this anti-encryption talk in Washington. Which isn't going away and appears to be gaining steam. We've previously talked about how RCS, which stands for Rich Communication Services, is the long-awaited successor to the carrier-provided SMS and MMS. We talked about RCS years ago.

Last week the folks at APKMirror got their hands on an internal build of Google Messages, the app, v6.2. The APK Insight team dug into it to see what appears to be coming with the next version of Google Messages. You can just pick up little hints from typical, like, strings in the code. The biggie that caught their attention was pretty clear evidence of end-to-end encryption being added to RCS Messaging.

So for some years now, as we know, it's been seen in RCS Messaging. It's been seen as a successor to SMS and MMS messages, and a potential competitor to Apple's kind of multimedia iMessage. However, one thing that iMessage and the whole Apple ecosystem offers that RCS could not was the ability to exchange privacy-protected messages that had the benefit of end-to-end encryption.

Of course, we'll need to wait for an official whitepaper before we see how all the various pieces fit together, you know, how Google has solved this problem, because that's not just a matter of the underlying crypto, which is pretty easy. We'll be talking about this more when we talk about what Zoom is doing in here in a minute. But if nothing else, Google has the significant advantage of being a late mover in this space, with many

examples of earlier efforts to guide their ultimate design. And they know crypto. So I would imagine they'll do it right when they choose to do it. And it looks like that's what they're choosing.

The APKMirror folks have found, in reverse engineering this latest v6.2 of Google Messages, 12 separate strings that make very clear reference to RCS-based encryption. I've got them numbered and called out here in the show notes. The first one, the string's name is "encrypted_rcs_message," and the string content is "End-to-End Encrypted Rich Communication Service message." So it's like it's...

Leo: That's pretty unequivocal, yeah.

Steve: Yeah. Then we have "send_encrypted_button_content_description." And the content is "Send end-to-end encrypted message." Okay, push that button, what do you think might happen? Then we've got "e2ee_conversation_tombstone" is a string name. And that string expands to "Chatting end-to-end encrypted with %s," which we know is the typical printf format for expanding a string in there. So the point is it would put up on the screen "Chatting end-to-end encrypted with Leo," and that's what it would - so it would declare that.

Then we get "metadata_encryption_status," whose content is "End-to-end encrypted message." Pretty self-explanatory. "e2ee_fail_to_send_retry_description," and then that says "Resend as chat." And that was one thing that the reverse engineers noted, and some of this text suggests, is that something about this requires more endurance. It's not exactly sure what. But, for example, there's a string named "encryption_fallback_title," and that string is "Send unencrypted messages?" And then there's "encryption_default_fallback_body," which reads "SMS/MMS texts aren't end-to-end encrypted. To send with end-to-end encryption, wait for improved data connection or send messages now as SMS/MMS."

So I guess the point is that it looks like cellular is able to do SMS/MMS, but you need an Internet connection separate from or on top of cellular in order to do the RCS encrypted messaging. Then they have "encryption_fallback_dialog_accept_button," and that button will be labeled "Send unencrypted." We've got "encryption_fallback_dialog_decline_button," which is labeled "Wait." Then "encryption_sent_fallback_body," which reads "SMS/MMS texts aren't end-to-end encrypted. To send with end-to-end encryption, wait until" something, this looks like a string has data connection. Oh, it's the person, like Leo, "has data connection or send messages now as SMS/MMS."

Second to last is, if this is proper, "etouffee_to_telephony_setting_title," and that says "Let other apps access end-to-end encrypted messages," which is interesting. And then, finally, "location_attachment_picker_send_encrypted_content_description." And that string is "Send end-to-end encrypted messages with selected location," and then the location gets filled in.

So anyway, there's been no public disclosure that these guys were aware of. I've certainly not run across any talk of Google Messages going encrypted. But it's very clear that there's some focus being given to that. I mean, this looks like it's done, and it's just a matter of them playing with it off the books for a while, working out the details. And I imagine at some point in the future we're going to be getting encrypted Google Messaging when you've got an Internet connection. Crazy the way the Internet works that way.

Okay. Oh, boy. We have the return of a much more worrisome StrandHogg, which is affecting all Android phones not running the latest Android 10. They're all vulnerable. So our listeners may remember that it was December of last year, earlier December, that StrandHogg 1.0 appeared.

The Hacker News wrote at the time, in December of 2019: "Cybersecurity researchers have discovered a new unpatched vulnerability in the Android operating system that dozens of malicious mobile apps are already exploiting in the wild to steal users' banking and other login credentials and spy on their activities. Dubbed StrandHogg, the vulnerability resides in the multitasking feature of Android that can be exploited by a malicious app installed on a device to masquerade as any other app on it, including any privileged system app. In other words," they wrote, "when a user taps the icon of a legitimate app, the malware exploiting the StrandHogg vulnerability can intercept and hijack this task to display a fake interface to the user instead of launching the legitimate application."

Okay, so that was then. Last week the same Norwegian cybersecurity researchers who found and obviously named StrandHogg being used in the wild unveiled details of an extremely critical successor vulnerability. And they've known of this for a while. The CVE is CVE-2020-0096. Again, 0096. The low number of that CVE suggests that it was assigned during the first few hours of the new year. It affects Android OS and allows attackers to carry out a much more sophisticated version of the StrandHogg attack. They of course called it StrandHogg 2.0. And this vulnerability affects all Android devices except those running the very latest version. And that's because they told Google about it at the beginning of the year, and Google was able to incorporate a fix for it into Android 10. But that's only 15 to 20% of Android devices, which leaves the other billion-plus Android smartphones currently vulnerable to this new attack.

I have a link to their formal vulnerability disclosure, and of course it's got sort of a Norse Viking logo thing on the page, Promon.co. And they said: "Promon researchers have discovered a new elevation of privilege vulnerability in Android that allows hackers to gain access to almost all apps." Which is one of the big differences. "Classified as 'Critical Severity' by Google, the vulnerability has been named StrandHogg 2.0 by Promon due to its similarities with the infamous StrandHogg vulnerability discovered by the company in 2019.

"While StrandHogg 2.0 also enables attackers to hijack nearly any app, it allows for broader attacks and is much more difficult to detect, making it, in effect, its predecessor's evil twin. Having learned from StrandHogg and subsequently evolved, StrandHogg 2.0 does not exploit the Android control setting 'TaskAffinity" - which is what 1.0 did - "which is what hijacks Android's multitasking feature and, as a result, leaves behind traceable markers." Meaning 2.0 doesn't.

"StrandHogg 2.0 is executed through reflection, allowing malicious apps to freely assume the identity of legitimate apps while also remaining completely hidden. Utilizing StrandHogg 2.0, attackers can, once a malicious app is installed on the device, gain access to private text messages and photos, steal victims' login credentials, track GPS movements, make and/or record phone conversations, and spy through a phone's camera and microphone." Clearly this is a huge win for the state-level targeted attack use case where they want to surveil somebody. If you don't have Android 10, this can be done to you, is what this amounts to.

Whereas StrandHogg 1.0 leveraged a vulnerability in the multitasking features of Android, this next generation 2.0 flaw is an elevation of privilege vulnerability that allows hackers to gain access to almost any app. Unlike 1.0, which was only able to attack a specific app one at a time because it needed to have that app named in an XML manifest

that allowed also those apps to be found in the Play Store, there's nothing visible in StrandHogg 2.

So they summed up in some bullet points: "StrandHogg flaws are potentially dangerous and concerning because it's almost impossible for targeted users to spot the attack. It can be used to hijack the interface for any app installed on a targeted device without requiring any configuration. It can be used to request any device permission fraudulently. It can be exploited without root access. It works on all previous versions of Android except Q, the latest. It does not need any special permission to work on the device. I mean, so this is really bad.

And of course, besides stealing login credentials through a convincing fake screen, the malware can also escalate its capabilities significantly by tricking users into granting sensitive device permissions while posing as a legitimate app. And it's much more difficult to detect because it's using code-based execution rather than, as I mentioned, needing a manifest to be bound into the app ahead of time, which made it much more visible. There's no effective and reliable way to block or to detect task hijacking attacks.

So the only thing that can be done until you are able to get a version of Android that has this patched, and hopefully those will be forthcoming, is to look for, I mean, to like be extra sensitive about behavior that doesn't seem quite right, that you wouldn't normally expect. For example, an app you're already logged into asking for a redundant login. If you know the app doesn't do that, then that would be StrandHogg 2 overlaying the app, trying to get you to log in again so that it could capture your login credentials for the app. Or permission pop-ups that do not contain an app name. I don't know why they couldn't be customized, but they're just suggesting that would be one to look for.

Permissions asked from an app that should not require or need the permissions it's asking for. Hopefully, everyone's always on the lookout for those. Buttons and links in the user interface that do nothing when clicked on, meaning that they presented sort of a partial mockup of a UI and didn't bother to make it complete. Or the back button does not work as expected, you know, because they're trying to trap you there and get you to go in the direction that they want you to.

So anyway, this exists. It's real. Google rates it critical. They fixed it in Android 10. But it's going to need patches in previous versions of Android. And hopefully those will be forthcoming. The problem is, as we know, lots of older devices are never going to be patched. And now that this has been disclosed, it is there forever. And it's clearly going to be the choice for high-level actors, state-level actors, in targeted attacks, to attack people. Maybe it'll become more widespread. If these StrandHogg 2.0 apps can get themselves into the Play Store, then it's not just targeted attacks, it's targets of opportunity.

So again, really a reason to only be using a smartphone that is actively receiving updates. Look at the panic that people experience when their desktop OSes are not constantly on the IV update feed, getting a constant drip of security fixes. We should not consider our phones to be any different. They are pocket computers. As we've talked about, many people are using them now in place of desktop or laptops because they've become so capable. You cannot be using it if it's not getting updates. I just think we need to have an industry-wide change in policy to require updates as part of consumer protection. To require updates as long as a phone is offering useful service, it needs to be receiving a flow of updates in order to fix these kinds of problems. Hasn't happened yet. We've talked about this before. But needs to.

The SHA-1 hash is finally going to be dropped from OpenSSH. The release notes from last Wednesday's OpenSSH update read: "It's now possible to perform chosen-prefix attacks against the SHA-1 algorithm for less than 50,000 USD. For this reason, we will be

disabling 'ssh-RSA' public key signature algorithm by default in a near-future release." In other words, OpenSSH will be finally shutting down SHA-1 as an available algorithm. They wrote: "This algorithm is unfortunately still used widely despite the existence of better alternatives, being the only remaining public key signature algorithm specified by the original SSH RFCs."

Now, to no one's surprise, the very real problem is that the world is full of aging network switches, low-cost embedded machines, many running ATMs and industrial control systems. They all contain older SSH servers that only speak SHA-1. This means that new systems that will be lacking support for SHA-1 will no longer be able to connect as they always could. It's truly a mess. It effectively means that much as we may wish to stop using SHA-1, we really cannot for those applications. Old OpenSSH-based clients will remain for use in connecting to those old servers. But those older clients will no longer be maintainable, since any updates to them would remove their support for SHA-1, which they need in order to connect to the old servers. Thus, embedded servers that cannot be updated are holding back updates to clients that would like the advantage of better security, but they can't update.

So I don't know what'll happen. I guess maybe people will start running two sets of clients. They'll have the clients that they can use to connect with more security, but they'll have to keep an old creaky SSH client around that still supports SHA-1 for those instances where they need to do administration to embedded devices that still only know SHA-1. Wow. It's a mess.

Speaking of messes, what happens when you fuzz USB? You find 26 new, previously unknown flaws, many of them critical, residing in our USB-supporting desktop OSes. We've talked many times about how important fuzzing is as a semi-automated means of uncovering previously unknown flaws in software. I would argue that those who are interested in exploring the idea of being a vulnerability finder for profit, probably the best way to start is with fuzzing. You've got to be a coder sort of by definition. So code up a fuzzer of something and turn it loose. And when something crashes, that's your next challenge, figure out what happened that crashed it and see if there's a vulnerability there that you can turn into a payday.

But in any event, the researchers who built this new USB fuzzing tool used it to uncover one bug in FreeBSD; three bugs in macOS, two of those resulting in an unplanned reboot and one which froze the system. They found four flaws in Windows 8 and 10. Those resulted in Blue Screens of Death. That's always, you know, where a flaw starts to become a vulnerability. And the greatest number of bugs and those most severe were found in Linux, a total of 18. 16 of those 18 were memory bugs having high security impact in various Linux subsystems - the USB Core, USB sound, and the network. One bug resided in the USB host controller driver, and the last was in a USB camera driver.

The cool thing is the researchers responsibly reported these bugs to the Linux kernel team, along with proposed patches, to reduce the burden on the kernel developers when fixing the reported vulnerabilities. And this is one of the powerful advantages of Linux being open source. Those finding a problem can directly examine the cause of the problem in the source and propose patches to repair the problem. This is not something that can be done with Windows or macOS, since those systems are closed-source mysteries. Of those 18 Linux bugs, the research team said 11 received a patch since their initial reports last year, so leaving seven outstanding. 10 of the 11 bugs also received a CVE designation because it was deemed serious enough. Additional patches are expected in the near future for the seven remaining issues.

The researchers said: "The remaining bugs fall into two classes: those still under embargo/being disclosed, and those that were concurrently found and reported by other researchers." So they plan to present their research at the USENIX Security Symposium

virtual security conference scheduled for August of this year, in 2020. And they've been preceded by other USB fuzzing tools. There was one called vUSBf. There's one, syzkaller, and usb-fuzzer.

So they're not the first to fuzz USB. But they feel that their tool, called USBFuzz, is superior to those that came before because it provides testers with more control over the test data, and it's portable to non-Unix-like OSes, so it'll run under Mac or Windows as the host of the tool, unlike any of the other tools. And it'll be released on GitHub as an open source project following their USENIX talk. If anyone is interested, I've got the repo link here, although at the moment it just takes you to a 404 error. They have not yet put up a page, but they will be.

SpinRite: Yesterday I reached the end of a surprisingly challenging bit of development for the SpinRite project. As we all know, SpinRite has always run on top of the FreeDOS clone of MS DOS. The SpinRite v6.x series, that is, .1, .2, and however many I do to basically catch it up with the technology change that's happened since 2004, will continue to do so, since this will let me get it out sooner to everybody who's waiting. But although that SpinRite will no longer use the BIOS, that's what all this next work is about, it does still use DOS. And DOS can still only boot from a BIOS-based system. And the FreeDOS people have stated - that's the only DOS that's still alive - they're never going to support UEFI.

So once the SpinRite 6 series are finished, my plan is to immediately remove SpinRite's dependence upon DOS so that it is able to boot natively as its own OS, essentially. And that will allow it to also boot from UEFI-based systems that will not boot, as I said, any form of DOS. One of the most frequently used words in that prior paragraph is "boot," which is where my efforts the past three weeks have been spent.

I now have a little Windows command line utility that can take a removable USB drive in any possible initial state - already formatted, unformatted, never formatted, formatted with some foreign thing like by the macOS with a GPT plus a FAT file system, whatever, with or without any defined partition, whether or not bootable, and replace it with a pure and perfect new bootable system. It runs on anything from Windows XP through Windows 10. I'm using only my own new partition and file system code, which I've just written, so that I'll be able to evolve this codebase in the future when it needs to be prepping a more complex bootable drive that can, for example, boot under either the BIOS or UEFI.

It was surprisingly tricky to get it working under every condition and circumstance. But thanks to the terrific testing of the gang who's hanging out over in the grc.spinrite.dev newsgroup, it appears as of yesterday that we are finally there. Once it's finished, that code, because right now I just quickly put it together as a command line EXE, but it's all Windows code. That code will be moved into SpinRite's boot prep GUI for easy use. And I spent this much time to get boot prep exactly right because everything in the future that I'll be doing will be using this code: the initial SpinTest drive benchmark, which will be freeware, and which everyone here listening to this podcast will be invited to play with; all the future SpinRite work; and also the Beyond Recall secure drive wiping utility. They're all going to be bootable.

The utility has even managed to repair a number of its testers' thumb drives that were either dead or thought to be dead or messed up in various ways so that they'd stopped working. So it brought them back to life. And since it also usually recovers a bit of previously unavailable and unused space on USB drives, I expect I'll also release it as a piece of freeware as a command line for removable USB drive formatting.

So anyway, that is done, and I will be returning to - we're going to do a little work now on using it to boot FreeBSD. Actually, it's doing that already, and it appears to be fully

functional. I just haven't turned my attention to it until it was time. And I've now got booting nailed. So I'm happy to have this technology, as I said. It's what we'll be using moving forward. And when you use SpinTest or SpinRite to create a boot USB, it'll just work. It handles all of the possible weird things that could go wrong for the user.

So from Zoom Communications we have Josh Blum, Simon Booth, Oded Gal, Maxwell Krohn, Karan Lyons, Antonio Marcedone, Mike Maxim, Merry Ember Mou, Jack O'Connor, and Miles Steele.

Leo: Couple of those names we know well because they came from Keybase.

Steve: Ah, exactly, exactly. And there is a reference to them in the doc. From Johns Hopkins we have well-known cryptographer and professor Matthew Green.

Leo: Wow, okay.

Steve: From Stanford University we have Alex Stamos.

Leo: Okay.

Steve: And we also have privacy expert Lea Kissner, formerly Google's global lead for privacy technology.

Leo: Pretty impressive.

Steve: Their names adorn the top of the 25-page detailed technical cryptography paper they collectively published on GitHub Friday before last on May 20th. That paper is simply titled "E2E Encryption for Zoom Meetings." And it begins by stating its purpose: "Hundreds of millions of participants join Zoom Meetings each day. They use Zoom to learn, among classmates scattered by recent events; to connect with friends and family; to collaborate with colleagues; and, in some cases, to discuss critical matters of state. Zoom users deserve excellent security guarantees, and Zoom is working to provide these protections in a transparent and peer-reviewed process.

"This document, mindful of practical constraints, proposes major security and privacy upgrades for Zoom. We are at the beginning of a process of consultation with multiple stakeholders including clients, cryptography experts, and civil society. As we receive feedback, we will update this document to reflect changes in roadmap and cryptographic design."

So as we've observed here on this podcast many times, across a great many different security challenges, in the end the weakest link in an end-to-end encryption system is the management of that system's keys. It all comes down to the keys. The task of actually encrypting and decrypting data at each end, given a shared key, has long ago been solved. How often are we talking about the way Apple manages iMessages keys, and how much does our U.S. Attorney General lust for some form of magical golden key? That's what he wants.

So Zoom's security team has properly determined that the first problem to solve, the first place to tighten things is with the system's key management. Once you have that, then the next weak link is identity, verifying that meeting participant identities are not being spoofed or forged. But it makes no sense to worry about that before you know that the resulting keys are safe.

So here's how Zoom describes the operation of their current system, the one they have now. They said: "Zoom Meetings currently use encryption to protect identity, data for meeting setup, and meeting contents. Zoom provides software for desktop and mobile operating systems, and embeds software in Zoom Room devices. In this document, when we refer to 'Zoom clients,' we include all of these various forms of packaging. Crucially, these are systems to which we can deploy cryptographic software. Zoom Meetings also supports web browsers through a combination of WebRTC and custom code. If enabled, Zoom meetings support the use of clients not controlled by Zoom, namely phones using the Public Switched Telephone Network (PSTN) and room systems supporting SIP and H.323," you know, IP phones technology.

They said: "In the meeting settings, as opposed to webinars, Zoom supports up to 1,000 simultaneous users." Webinars, I didn't realize, Leo, they could be 50,000 people in a webinar. Whoo. "When a Zoom client gains entry to a Zoom meeting, it receives a 256-bit per-meeting key created by Zoom's servers, which retain the key to distribute to participants as they join. In the version of Zoom's meeting encryption protocol set for release on May 30th" - so that has already happened; that was, what, last Saturday, three or four days ago - "this per-meeting key is used to derive a per-stream key."

Okay. So essentially what they originally had, when we first started talking about Zoom, when coronavirus hadn't yet happened, was a single key which the server created, which was simply distributed to all the clients who joined, to use that for encryption and decryption. So, yes, it was encrypted. Technically it was end to end. But practically, the server made the key. So it was no mystery to anybody within the Zoom infrastructure. One of the things that is clearly necessary is to arrange to blind Zoom's infrastructure to the key being used by the clients to communicate so that the infrastructure is simply transiting pre-encrypted data that it is unable to see into.

But they made an interim change on Saturday, May 30th. The per-meeting key, which used to just be one, is now being used to derive a per-stream key by combining the per-meeting key with a non-secret stream ID using an HMAC. Each stream key is used to encrypt audio/video UDP packets using now AES in GCM mode. Remember they were using electronic codebook, which is not very secure, or at least it leaks information about the plaintext, so GCM is way stronger, with each client emitting one or more uniquely identified streams. Those packets are relayed and multiplexed via one or more Multimedia Routers, they call them, MMRs, Multimedia Routers, within Zoom's infrastructure.

The MMR servers do not decrypt these packets to route them. There's no mechanism to re-key a meeting. Videoconferencing systems in which the server relies on plaintext access to the meeting content to perform operations such as multiplexing would be exceptionally difficult to secure end to end. You know, you're asking for something that isn't end to end. In this design, we take advantage of the fact that the Zoom servers do not require any access to meeting content, allowing end-to-end security at exceptionally large scale.

Then they put in some caveats, of course. If telephones or IP phone clients are authorized to join, in that case the MMR, the multimedia router, provides the per-meeting encryption key to specialized connection servers within Zoom's infrastructure. These servers act as a proxy. They decrypt and composite the meeting content streams in the same manner as a Zoom client and then re-encode the content in a manner

appropriate for the connecting client. Zoom's optional cloud recording feature works similarly, recording the decrypted streams and hosting the resulting file in Zoom cloud for the user to access. In the current design, Zoom's infrastructure brokers access to the meeting key.

So anyway, so they're basically saying, if you have a client that is unable to have the capability of doing the encryption/decryption, if it's a dumb client like a telephone or an IP phone that all it can do is receive and transmit standard data, then they will proxy for that client. And then the link to the client is unencrypted because the client doesn't support encryption.

They finish: "This current design provides confidentiality and authenticity for all Zoom data streams, but it does not provide true end-to-end encryption as understood by security experts due to the lack of end-to-end key management. In the current implementation, a passive adversary who can monitor Zoom's server infrastructure and who has access to the memory of the relevant Zoom servers may be able to defeat encryption. The adversary can observe the shared meeting key, derive session keys, and decrypt all meeting data." Again, you've got to be at Zoom to do that. But still.

"Zoom's current setup, as well as virtually every other cloud product, relies on securing that infrastructure in order to achieve overall security. End-to-end encryption, using keys at the endpoints only, allows us to reduce reliance on the security of Zoom's infrastructure." And that's where they will be heading.

So they divided the task of getting from where they are now to where they want to go into four successive phases. And this 25-page document, I saw you had the top of it, Leo. If you scroll down about, oh, not that far, like a third of the way, you start hitting all the crypto equations. And it's like, okay.

Leo: Yeah, it's cool.

Steve: So, I mean, it's all there. But it's really not necessary. I mean, essentially, we know how to do crypto now. There's no way for me to discuss the detailed design of the various algorithms on an audio podcast. And there's a large amount of it. But that doesn't really matter because, when it's done correctly, it works. And everybody who's there knows how to do it correctly. The overall architecture is the key, and that can be usefully characterized.

So the four phases are client key management, identity, a transparency tree, and real-time security. The transparency tree is what the Keybase guys brought to the party, as we'll see in a second. So let's look at what they've explained about each of those briefly. The first phase, they said: "In the first phase we will roll out public key management, where every Zoom application generates and manages its own long-lived public/private key pair. Those private keys are known only to the client. From here, we will upgrade session key negotiation so that the clients can generate and exchange session keys without needing to trust the server." And we know we've talked often about how that's done. Diffie-Hellman key agreement, where in total view you're able to look at data going in each direction, allows the endpoints to arrive at a mutually known key, but nobody looking at the data going back and forth is able to derive anything of use. So that we're able to do. They'll be adding that in this first phase.

They said: "In this phase a malicious party could still inject an unwanted public key into the exchange." So that's a good point. This is not, because authentication is still weak, the authentication is Phase 2, identity. So as we know, public key crypto assumes you're exchanging with the right person. If a man in the middle does a double exchange with

each endpoint, then they don't know that they haven't exchanged with the endpoints themselves. They've exchanged with a man in the middle who now is able to essentially negotiate separate keys with each of them and is able to form a bridge through which to decrypt in the middle. So again, they're taking this a step at a time, strengthening the infrastructure as they go.

They said: "We offer meeting security codes as an advanced feature." And we've talked about this, too, in the past. "So motivated users can verify these public keys. The security to be achieved here will approximate those of Apple's FaceTime and iMessage products." We talked about this. I think both Threema and - there have been a number of messaging systems where you're able to, over the channel, you're able to say I'm going to - let's verify our fingerprints. And so each of the endpoints is able to verify the fingerprint, which cuts out, or at least would immediately alert to any man in the middle. At the moment that's a manual process. Thus they said "motivated users." They'll be making it automatic.

So they said: "The primary improvement in Phase I is that a server adversary must now become active, rather than passive, to break the protocol. In Phase I, we will support native Zoom clients and Zoom Rooms. We will not support web browsers, dial-in phones, and other legacy devices. There also will be no support for Join Before Host, Cloud Recording, and some other Zoom features." Anyway, so I had a note here. I said so Phase I takes them from the current encrypted system where the individual client encryption keys could be captured by a passive observer at the Zoom server to a system where Diffie-Hellman secure key establishment is used to blind a passive observer to the client's session keys.

Phase II, identity. They wrote: "In the first phase, clients trust Zoom to accurately map usernames to public keys." And of course you're also trusting that there's no active attacker. They said: "A malicious Zoom server in theory has the ability to swap mappings on the fly to trick participants into entering a meeting with imposters. In Phase II, we will introduce two parallel mechanisms for users to track each other's identities without trusting Zoom's servers. For users authenticating to Zoom via single-sign-on, we will allow the single-sign-on identity provider to sign a binding of a Zoom public key to a single sign-on identity, and to plumb this identity through to the UI. Unless the single-sign-on or the identity provider has a flaw, Zoom cannot fake this identity."

So in other words, that's saying that the UI will be able to attest to the identity of the person on the other end by verifying the signature of the other user's single-sign-on identity provider in order to say, yup, there's no way this isn't the right guy, at least as far as single-sign-on goes. Or, second, we allow users to track contacts' keys across meetings. That is, the key should be static over time because it's created once in the client. They said: "This way, the UI can surface warnings if a user joins a meeting with a new public key."

And again we've seen, for example, Signal. You can turn on an alert to be alerted if the user's key ever changes. It's like, okay, that shouldn't happen. That could be an active attack. So this tightens up security in a way similar to other semi-identity-managed messaging systems, like I mentioned Signal and Threema, by remembering previous keys and letting you know if they changed.

Phase III, a transparency tree. They say: "In the third phase, we will implement a mechanism that forces Zoom servers and single-sign-on providers to sign and immutably store any keys that Zoom claims belong to a specific user" - basically Zoom becomes a clearinghouse - "forcing Zoom to provide a consistent reply to all clients about these claims." In other words, in Phase II it was up to the user to verify the key of the other end. Now Zoom is going to be involved in that. It will maintain, very much like a

certificate transparency log, it will maintain a log of all the keys of all the clients - that is, the public keys, not the privacy keys - and proactively take measures.

They said: "Each client will periodically audit the keys that are being advertised for their own account and surface new additions to the user." That is, if something new appears for their account, it's like, wait a minute, somebody is impersonating you. "Additionally, auditor systems can routinely verify and sound the alarm on any inconsistencies within their purview. In this scenario, if Zoom were to lie about Alice's keys, say in order to join a meeting which Alice is invited to, it would have to lie to everyone in a detectable way."

They wrote: "We will obtain these guarantees by building a transparency tree, similar to those used in Certificate Transparency and Keybase. During this phase we will also provide the capability for meeting leaders to 'upgrade' a meeting to end-to-end encrypted after it has begun, provided that all attendees are using the necessary client versions, and incompatible features are not in use. Such incompatible features include dial-in phones, IP phone, room systems, and cloud recordings. Meetings that cannot be upgraded will have the option grayed-out. We will also reenable 'Join Before Host' mode," allowing people to show up before the host logs in.

And finally, Phase IV, real-time security. Consider this hypothetical attack against the Phase III design. A malicious Zoom server introduces a new "ghost" device for Bob, a user who does not have their identity provider vouch for their identity - I timed that wrong - a new ghost device for Bob, who is a user. Bob is a user who does not have their identity provider vouching for their identity. Thus some of the protections that would be available otherwise are not present.

The attacker, using this fake new device, starts a meeting with Alice. Alice sees a new device for Bob, but does not check the key's fingerprint. After the fact, Bob can catch the server's malfeasance, but only after the attacker tricked Alice into divulging important information. The transparency tree encourages a "trust but verify" stance, where intrusions cannot be covered up.

In Phase IV we look to the future, where Bob should sign new devices with existing devices, using a single-sign-on identity provider to reinforce device additions, or delegate it to his local IT manager. Until one of these conditions is met, Alice will look askance at Bob's new devices. In other words, if a new unsigned device, one that has not been signed by an existing trusted device, is used, the UI that Alice is seeing will be saying, look, "Bob" is not following the rules here. We're making no allegations or attestations that this is actually Bob. You be careful what you say because it may not be, and you've got to have Bob explain why he's using a device that another one of his devices hasn't permitted and authorized and signed.

So they conclude, saying: "Across these next four phases of security architecture design, we will end up with a state-of-the-art multi-party teleconferencing system which provides robust guarantees of participant identity, takes all key-awareness away from the Zoom infrastructure, and is able to proactively alert any Zoom user when something looks wrong."

Oh, and one thing I noted that I did want to point out about the crypto details was this paragraph. They said: "In an ideal world, all public key cryptographic operations could happen via Diffie-Hellman over Curve25519" - my favorite chosen curve - "and EdDSA" - that's Edwards DSA - "over the Edwards25519 curve." Which of course are the technologies I chose for SQRL, what, seven years ago, eight years ago.

"Relative to the others," they're writing now, they're saying: "Relative to others, this curve and algorithm family have shown a consistent track record for resilience to common cryptographic attacks and implementation mistakes. These algorithms are

currently in review for FIPS certification; but, unfortunately, are not yet approved. Therefore, in some cases, like government uses that require FIPS certification, we must fall back to FIPS-approved algorithms like ECDSA and Elliptic Curve Diffie-Hellman over the P-384 curve. These protocols we use have support for both algorithm families, and for now double-up all public key operations to eliminate error-prone if/then/else branching. Once certification succeeds, we can safely 'no-op' the operations over P-384 curve."

And I thought, what a clever solution. What they're saying there is, since the operations where these are being used are inherently not time sensitive, they're one-time setup things, and they're only being used to establish keys and identities, why not use both the 25519 and P-384 curves in series? Then, once 25519 has been shown to be sufficient, simply drop the P-384 out of the chain by no-opping it, and you're left with the curve that you trust, and you don't need to worry about implementation mistakes in P-384 mucking up the whole crypto.

So nice piece of engineering. I think what they're doing makes total sense. And it's clear to any observer that Zoom is absolutely serious about offering users continuing the same ease of use, but incrementally taking increasing responsibility for making the system much more bulletproof behind the scenes. End users won't notice any difference. It will continue to operate just as it did before. But they will be notified when anything screwy might be happening. So hats off to these guys and to this team that has put this together.

Leo: Do they give an ETA for when this will be implemented?

Steve: There's no timing at all.

Leo: This is just a proposal, then.

Steve: Yeah.

Leo: But of course you've got the best crypto guys in the world there, so I'm sure it's all well thought-out and very sound.

Steve: And they don't have to invent anything. All of this is just applying well-known crypto to solve the particular problem that they have.

Leo: I mean, I can see it if it's a one-to-one call. That's trivial. That's like messaging. That's like Signal. But I thought it was difficult to do a one-to-many call or many-to-many call because you had to, at the server point, decrypt; right?

Steve: I'm stunned, Leo. Maybe that's why they specifically said not for webinars, but for meetings up to a thousand. You're talking about a 1,000-way encryption. Because you've got to encrypt every key pair to every other user. It's like, yo, ho. Boy.

Leo: So at the control point there's no need for the control point to have a decrypted version. They can route it all based on the crypto keys. And that's all they really need; right? Yeah.

Steve: Right.

Leo: Yeah, that makes sense. It's just a lot of keys. It's a lot of management.

Steve: Well, it's a lot of streams. It's not one stream going up just to headquarters and then being sent back out.

Leo: Oh, that's a good point, yeah.

Steve: It is a thousand streams from every user, or 999 from every user to all other 999.

Leo: I don't know if that's how they do it normally anyway, or if they try to save bandwidth by aggregating and doing broadcasts?

Steve: I think it's the way they've been doing it, interestingly.

Leo: Yeah, I think it is, yeah.

Steve: Because they have done end to end, but it was just with keys that the server could know about.

Leo: Right. Well, and that's what I thought is that the server needed to know about it to do the routing. But I guess not.

Steve: Yeah. Well, the good news is symmetric crypto is what you're using with the key. And symmetric crypto is fast.

Leo: Yeah, it's a lot easier.

Steve: And using AES-GCM doesn't slow it down at all.

Leo: Right, right.

Steve: So it's all, you know, and now Intel chips have AES acceleration in them.

Leo: Right.

Steve: They actually have three or four instructions that, like, make the most time-intensive bit-twiddling bit just instant.

Leo: That's cool. That's really good.

Steve: Yeah, very.

Leo: Steve, as always, fascinating stuff. This is a show everybody needs to listen to every week. We do Security Now! on Tuesdays, 1:30 Pacific, 4:30 Eastern, 20:30 UTC. You can watch live. I mean, you don't need to. But if you wanted to, or you're just around on a Tuesday afternoon/evening, just go to TWiT.tv/live. That's where the audio and video streams are aggregated. You can pick one you like. They are not encrypted. You don't need a key. They're just wide open. We want everybody to see them. If you're doing that, join us in the chatroom, also not encrypted, at irc.twit.tv. That's a great place to be.

If you're going to watch after the fact, there's a couple places to get your shows. You can get them from Steve. He's got 16Kb audio for the bandwidth-impaired, 64Kb audio, and he has transcripts. That's the one place you can get those really well-done transcripts from Elaine Farris. That's GRC.com. While you're there, check out the place. There's lots of free stuff. It's an amazing site. Easy to get lost, fall down a rabbit hole. So plan some time as you visit GRC.com. Steve is also on Twitter at @SGgrc. So if you want to message him, that's the best place to do it. You can DM him. His DMs are open, @SGgrc on Twitter. Or leave a feedback at the feedback form on the website, GRC.com/feedback.

We have audio and video at our site, TWiT.tv/sn. You can also see the show on YouTube. Best thing to do would be to subscribe. Get a good podcast app. There's lots of them out there, not labeled TWiT, just things like Pocket Casts and Stitcher and Slacker and Overcast, and Google and Apple have podcast apps. And subscribe. Just say Security Now!, I want it every week. That way it'll be on your device, and you can listen at your convenience, whenever you get a moment, a free moment.

Thanks, Steve. Have a wonderful week. Stay safe and healthy. We'll see you next week on Security Now!.

Steve: Right-o. Thanks, buddy.