Android "Q"

Description: This week we look at a widespread problem affecting all WhatsApp users, many interesting bits of news arising from last week's Google I/O 2019 conference, a worrisome remotely exploitable flaw in all Linux kernels earlier than v5.0.8, the just released hours ago new set of flaws affecting all Intel processors known as ZombieLoad, a bit of miscellany, and some odds and ends. Then we take a deep look into the significant security enhancements Google also announced in their next release of Android: Q.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-714.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-714-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. He's going to do a rundown of some of the nice new security features in Android Q. Some rare praise from Mr. G. for Mr. Google. We'll also talk about that WhatsApp flaw. A new attack on Intel processors called ZombieLoad. I love the name. And some very nice features coming to Google Chrome soon. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 714, recorded Tuesday, May 14th, 2019: Android Q.

It's time for Security Now!, the show where we cover your privacy, your security, all that stuff, with this guy right here, this cat. This guy is all about security, Mr. Steve Gibson of the Gibson Research Corporation. Hello, Steve.

Steve Gibson: Mr. Laporte.

Leo: How are you today?

Steve: Great to be with you once again for Episode 714. This is the second Tuesday, as I have been saying we would get to, as close to the middle of the month as we're able to have a Patch Tuesday. I have no idea what's happening on Patch Tuesday because everything else is...

Leo: Wasn't last Tuesday Patch Tuesday?

Steve: No, because Wednesday was the 1st, so we missed it by one day. Which makes this the second Tuesday of the month. So I did see that there's lots going on. We'll cover that, if there's anything particularly interesting, next week. In the meantime, everybody knows don't wait too long because lord knows what just got fixed.

Leo: Oh, yeah. I see cumulative update for .NET framework for Windows 1809, security update for Flash Player, cumulative update for Windows 1809, malicious software removal tool. Yeah, so it's the usual bundle of goodies.

Steve: Oh, yeah, the usual suspects. I started out with - let's see. Oh, I was going to talk about a forthcoming change in Chrome's policy for cookies, which really represents some nice forward motion. But then I started looking more closely at last week's Google I/O announcements of the security, well, security improvements - I was going to say security and privacy, but, no, just security - in Android's forthcoming Q release. And I was so taken by them and so impressed by them that I thought, okay, that's what we're going to end the podcast talking about is I want to spend some time looking at the forward movement over time and what I consider to be a tremendous and tremendously useful focus that Google has had on Android. I mean, they don't really have to. But I guess they can because they're Google, and they've got resources exuding out of their pores. And so why not really make it as good as they can? And, boy, it's - where are we? We're at Q. Well, okay. I don't know what happens when we run out past Z. But of course we don't even know which dessert Q is going to be. So that's still the pending question, I think.

But anyway, lots to talk about. We've got a widespread problem affecting all 1.5 billion WhatsApp users. Not super worrisome because it's going to be used in targeted attacks, but the back story is interesting. We've got, and I heard you talking about it a little bit on MacBreak Weekly, we have many interesting bits of news arising from last week's Google I/O 2019 conference surrounding the changes being made to Chrome. And since Chrome is the most used browser, and it's based on Chromium, that now, of course, as we know, Microsoft has picked up with their next version of Edge, this becomes extra interesting.

We've got a worrisome remotely - listen to this, children - remotely exploitable flaw in all Linux kernels earlier than 5.0.8. It's not the end of the world because it's very difficult to do. But it's in there. So end users, again, aren't going to be in a huge problem because it's a network-level exploit. But that does mean that Internet-facing Linux servers need to get themselves updated because what's difficult to do now becomes script-kiddie fodder six months from now. So you don't want to be caught with your kernel down.

We have also the just-released hours ago, I mean, like this morning it came out of embargo, it's from some people, some researchers we've covered extensively. They were the guys that did the Rowhammer attacks earlier. Back last March, not this most recent one, but still a fond memory, March 2018, in the middle of the month, they found another way of exploiting the microarchitecture of all Intel chips since 2011. And they said, uh, Intel? Anyway, so the news of that has been embargoed until this morning. So it's called ZombieLoad.

Leo: Oh, geez. That's a great name, man.

Steve: That's right, ZombieLoad. And there's a 15-page research paper that I have not yet had a chance to dig into because it just came out of embargo. So but I have a sense for it, and we'll talk about that. We've got a bit of miscellany, some odds and ends, and then we're going to take a deep look at the really significant security work that Google

has been doing on Android. And I conclude by saying that this is becoming a serious asset for humanity.

Leo: A serious asset for humanity.

Steve: Unlike the microwave that is our Picture of the Week that is a serious impediment to dinner.

Leo: You want me to show that now? "Please wait while we update your system to the latest firmware. When it completes, you'll see the screen go blank as it restarts. It may take up to 30 minutes." I want my Hot Pocket. You think that's real?

Steve: I think it's real. It looks like, well, it was tweeted to me by one of our listeners who said, "I've got to wait until my oven installs new firmware before I can make dinner." And he said: "#TheFutureIsNow," meaning, uh, yeah.

Leo: All right.

Steve: So it's not crucial unless you happen to be a human rights activist attorney-like person who might be subject to targeted attacks. But what came to light is that there was a flaw in all mobile versions of WhatsApp for both Android and iOS, both the standard and the business editions, as well as the Windows phone, which leveraged a buffer overflow in the VoIP, the Voice over IP protocol stack, which allowed a remote code execution when specially crafted packets were received by the WhatsApp app.

And what this meant was that, even without answering your phone, a remote code execution vulnerability could be leveraged to install, and it was being used to install, some very potent spyware known as Pegasus which is produced by an Israeli, I don't want to call them "malware maker." It's the NSO Group that was just purchased for a billion dollars. So they're a legitimate organization. They sell their stuff, they say, only to legitimate law enforcement agencies, yet some of their attacks have been spotted being misused. That is, being used for purposes that they claim they would not sanction. On the other hand, you wonder whether an organization like this doesn't probably look the other way and say, well, you know, they told us they wouldn't do that.

Anyway, there are updates to WhatsApp. This came to light last Friday. And the communications from WhatsApp and Facebook has been a little bit confused and muddled. There was some statement that said that it was fixed on Friday. But then another group within Facebook said that they changed the WhatsApp servers to mitigate this on Friday, but that just yesterday there was a new release for all of the WhatsApp code across the board. So anyway, it's good that this was found. We don't know how long it has been there, and who has been a victim. But it did let me know that there are 1.5 billion instances, 1.5 billion installs.

Oh, and I meant to say you do not even need to answer the incoming call. So the idea would be that the perp can target you by your WhatsApp, and you don't even have to be there. They're able to turn on the camera, rummage through your device, listen to what's going on. I mean, basically it's installing this Pegasus advanced system penetration malware which is able to jailbreak and/or root any targeted mobile device. So that's been shut down. We don't know what other tricks these people have up their sleeve. It always

seems that as many of these things as we foreclose, new ones are being found. But at least this is one fewer that is in operation.

Many things were announced last week at Google's I/O conference. And several of them, as I mentioned, I want to talk about Android Q at the end of the podcast. But there was an interesting change that has been announced and will be forthcoming two versions of Chrome from now, which represents a difficult, yet useful change to the handling of cookies. It's difficult, I think, because any change to something as ubiquitous as the way cookies are handled in our browsers is going to be difficult.

Yet there is a push, and I'm glad to see it, to sort of fix the problems, the acknowledged problems with cookies. Google has a person there who's been sort of on the forefront of this, spearheading the effort, named Mike West. And last Tuesday, on the 7th, the IETF Network Working Group Internet Draft titled "Incrementally Better Cookies" was published. And he recognizes that incremental is all we can get, but incremental is better than nothing.

So let me explain all of the background here because there's a lot going on. So in the introduction to this, Mike writes: "The HTTP State Tokens" - now, that's something we had never talked about before. That's like the dream is to do something other than cookies for HTTP state. As we've talked about, the whole reason that Netscape originally introduced a cookie to HTTP queries was because the way the browser and the server works, it's a stateless interaction. The browser makes a query. The server answers the query. And that's it. And then the browser makes another query. And then the server answers that query. But there's nothing to tie those events together.

So, for example, there was really no way to have the notion of a session, like a persistent session that spanned queries. There was no notion of a way to log into the server. All of this, thanks to Netscape, we got with cookies. Unfortunately, we got a lot that we didn't bargain for with cookies, as well. As I have often bemoaned, the fact that they were never meant to be used in a third-party context, only in a first-party context - and of course it's famously the third-party context that has created tracking. But on the flipside, it's created revenue which fuels organizations like Google, which is able to produce good things and is, over time, I really do feel, working to improve some of this foundation that we have.

So Mike says: "The HTTP State Tokens proposal aims to replace cookies with a state management mechanism that has better security and privacy properties." Arguably, cookies have neither. He says: "That proposal is somewhat aspirational." He says: "It's going to take a long time to come to agreement on the exact contours of a cookie replacement, and an even longer time to actually do so." So everybody here is being realistic about this.

He says: "While we're debating the details of the new state management primitive, it seems quite reasonable to reevaluate some aspects of the existing primitive: cookies." He says: "When we can find consensus on some aspect of HTTP State Tokens, we can apply those aspirations to cookies, driving incremental improvements to state management in the status quo." Meaning move these things forward.

So he writes: "Based on conversations at the HTTP Workshop 2019 and elsewhere, I'd suggest that we have something like agreement on at least two principles." So two things have emerged. He says: "First, HTTP requests should not carry state along with cross-site requests by default." And when I read that, I have to say I did a double-take. It's like, what? HTTP requests should not carry state along with cross-site requests by default, which is to say, okay, third-party cookies should not be. Which would, like, what? Anyway, and then the second thing is "HTTP requests should not carry state over non-secure channels."

Okay. So that's a little bit less aggressive hope, the idea being, I mean, for example, and we've talked about, back in the day of Firesheep, it was because cookies that you negotiated over a secure channel, when you were doing username and password back and forth, those cookies then continued to persist on the nonsecure transactions with, for example, Facebook and other websites. And because they were nonsecure, you could sniff them and then impersonate users. It's hard to imagine that it wasn't that long that was happening. But we fixed that by switching over to HTTPS and TLS connections. Everything is encrypted now on all major websites. And it's possible to mark a cookie with the SECURE tag, which will tell the browser "never send this except over a secure connection" in order to prevent its leakage if someone asks for a nonsecure asset from the same server.

So those are the two things, the two principles he's suggesting that they have something like an agreement on. So but the one that really brought me up short was this notion of not carrying state for cross-site requests. So traditionally cookies have had two tags that they could be set with. One was HTTPONLY, and the other was SECURE. I just talked about the SECURE tag. If a cookie is tagged as SECURE, the browser knows never to send it with a nonsecure query. The other one is HTTPONLY. What that says is only allow the cookie to be seen over HTTP. And what that means is specifically blind any scripting.

So the idea being is keep this away from JavaScript because we know that there have been ways to inject JavaScript into web pages where we didn't expect it to be injected. And if malicious JavaScript got onto a page, and it was able to query the cookie, and in fact JavaScript is normally able to query cookies that it has potentially - like it's supposed to have access to. Like if malicious JavaScript got into the domain of a secure cookie, it could see it. So flagging the cookies, so when the server sends the cookie to the browser, if it marks it, tags it as HTTP only, then the browser won't let any script running even on that site's own page. The web server is saying, I don't even want our own script to be able to see these cookies. Only send them back with queries so that we have state maintenance, and for no other purpose. Anyway, so those have been the two existing tags.

Well, it turns out that there is another newer one which is already supported by all of our browsers except IE. So all of the significant browsers - and I don't know, maybe Microsoft will update IE, or maybe they're just going to leave it at 11 and never push it any further. I don't know. And that tag is SAMESITE. SAMESITE is optional, as are these other two, HTTPONLY and SECURE. And it has, if it's not present, which is the normal case, it has one meaning, and then it can be explicitly given three different meanings. So here's what they are. This SAMESITE tag can be set to None, to Lax, or to Strict. If it's not set, then previously, or I should say right now, if it's not set then that's the same as None. That is, if it's not present at all.

What Google is proposing to do in, I think we're at Chrome 74, and so this will be appearing in Chrome 76, is they're going to change its "not present" meaning from None, which it's always been until now, to Lax, which is more strict than None, but less strict than Strict. Okay. So here's what's happening. If the cookie is tagged as being Strict SAMESITE, then that says that it should only be sent to the - it should only be sent back to the server in a first-party context. That is, essentially, do not allow this cookie to be sent in a third-party context, meaning if any asset on another site refers to the domain that set the cookie, for example an image, for example.

So Site B shows an image that it has pulled from Site A. The normal behavior would be for the browser that has cookies for Site A to return those cookies along with its query for this image from Site A. But if that cookie has been received with a SAMESITE set to Strict, the browser will know not to do that, not to send cookies out for peripheral assets, you know, images and other widgets, to third-party servers. So, and that's an arguably useful security benefit because there are a number of ways, like cross-site request

forgeries, that malicious script and queries can break this cookie protection. So that provides some protection from that.

So setting the SAMESITE to Strict means, essentially, restrict this cookie for first-party access only. If it's set to Lax, which it currently would have to be explicitly set for, not two Chrome versions from now but at the moment, if it's set to Lax, then SAMESITE cookies are also withheld on cross-site subrequests, like we were just talking about, like images and things. But not if the user navigates to a specific URL from some other site.

So, for example, by following a link, if the user clicks a link, then that cookie that the browser has would be sent with the link. So, for example, to immediately identify the user as having a logged-on session with a site that the user is going to. So that's the so-called Lax handling. So passive things are still blocked, but links are still going to be - will be honored. So but that also says that, under strict handling, a URL click would not provide the cookie. It wouldn't be until you actually are on the site and you have a page that then cookies would be first-party. And of course the default behavior is either it's not set, or you can say SAMESITE=None.

Okay. So what Google has announced they're going to do, they announced this last week, is in two releases from now they're going to change the default "no specification," that is, where cookies are set with no tag at all, no SAMESITE specification, they're going to change that from the way it has always been, which is to say, unless we're told to make it strict or lax, we're going to assume it's no protection. They're going to change that to Lax, which means suddenly all of the passive queries for things like images, which of course are web beacons, right, I mean, those are the famous little one-pixel queries that have always been used for tracking, those all go dead. They all go silent two versions from now.

So essentially Google is saying to all of the organizations and all of the tracking systems that exist, if you want to continue receiving cookies from all of the debris that you spread out all over the Internet, you need to start switching those cookies to explicitly say SAMESITE=None so that they will continue to be - so the users' browsers will continue sending back the cookie content for the domains they have in a third-party context. And that's a huge change. I mean, that's significant. I'm, like I said, when I read that it was like, whoa, okay, really. So again, props to Google for moving this forward.

Google has also said at the announcement time that what this does is it serves to mark cookies as being explicitly third-party, that is, right now there's no marking. As I have always said, this is sort of a mistake that this was allowed to happen this way, that has been well leveraged. And of course it glues the whole Internet together these days, so it's not like it's going to go away. But what this does is it marks cookies as explicitly intended for third-party access, which then allows Google to start doing some instrumentation. They've said they're wanting to experiment with showing cookies to users, giving users more explicit control over this.

And so requiring tracking cookies to essentially declare themselves as being for that purpose is an interesting piece of instrumentation that Google hasn't had until now. And again, as we've seen, Google tends to be the leader in this. They can afford to be because they're the majority browser on the Internet. And this is all in Chromium, so presumably Edge will inherit this. And I would imagine that Firefox, with the Mozilla team, will probably be following along. Everybody understands the SAMESITE tag except for IE. And so it's a trivial change, essentially, to just change the "there's no explicit statement about this, so we have been considering it as being promiscuous" with None, no SAMESITE exclusion. We're just going to change how we handle that unless we're told otherwise.

So I think what will happen is we'll quickly see, I mean, it doesn't foreclose anybody, it doesn't prevent anybody from immediately marking their cookies as SAMESITE=None. But it means they're going to have to do so if they want things like passive images on web beacons and so forth to be able to be used for tracking people. They'll have to be explicit about it. And so I think that's all for the best. And again, props to Google for pushing this.

They've also announced that they're going to do some anti-fingerprinting in Chrome. We don't know yet exactly what that means, what they're going to be doing to change. We've talked about fingerprinting in the past a number of times. Fingerprinting is - there's no way to consider this other than being sneaky, a sneaky attempt to track users in a way that was absolutely and definitely never intended. You could argue that tracking users with a cookie is, well, okay, it wasn't intended, but your browser received a unique token from a website. And now, as you go other places, and that website is somehow able to keep some content on those other sites that refer back to it, well, then it gets that unique token, and that's where tracking comes from.

Not so with fingerprinting. Fingerprinting uses a whole set of sort of obscure, never intended for identification things, like the user-agent header that the browser sends back with every query. We've talked about that the last couple of weeks because user-agent headers have been, you know, Edge is changing what it declares itself to be, depending upon which sites it's visiting. Those user-agent headers often have a bunch of version numbers in them. And while many browsers will have the same version numbers, many more other browsers will have different version numbers.

So although the user-agent header can't be used to uniquely identify a single browser, what it can do is, in the whole set of browsers in the world, it can immediately create a subset of those that that one browser occupies based on its unique, well, not its unique, it's based on a user-agent header that it shares with a subset of the world's browsers. And then, for example, it turns out that different web browsers issue their headers, their query headers in slightly, subtly different sequences. Like there's date and user-agent and itag and a bunch of other standard headers. Turns out that different browsers just send them out in a different order, since the order doesn't matter to the HTTP protocol. So that's another little signal that somebody who's trying to fingerprint a browser can use in order to further subdivide the set of browsers into a successively smaller set.

And we talked about, for example, it's possible to enumerate the fonts that are installed in a computer. Once again, many computers will have exactly the same set of fonts installed. But then again, many computers will have different sets of fonts installed. So you can sort of do all of these overlapping Venn diagrams and, with a surprising amount of specificity, narrow a browser down to, not probably to one, but to many fewer than all in the world, in a way that was entirely unintended by anybody. But once again, as we've seen, people are clever who are motivated to be so.

So Google announced last week that they're going to be blocking certain types of fingerprinting. We don't know what that means. Maybe they're going to randomize the order of the fields in the user-agent. Maybe they're going to randomize the sequence of headers in their queries. Who knows what they're going to do? But they're going to do something. The first major browser which really worked hard to block this fingerprinting technology was the Firefox version that Tor was using. Of course, if anywhere you would want to not have your browser, which you're using through the Tor anonymizing network, to be basically allowing you to be whittled down from anyone in the world to a tiny little subset of possible users based on fingerprinting.

So anti-fingerprinting technology is relatively mature. There's been a lot of work done to thwart the deanonymization of users, thanks to the work that the Tor version of the Firefox browser has adopted. So we don't know yet what it's going to be. I'm looking

forward to finding out. I imagine in time, once they finally bring this to us, we'll find out what's going on. As Google phrased it when they were explaining this, they said: "Because fingerprinting is neither transparent nor under the user's control, it results in tracking that doesn't respect user choice." They said: "This is why Chrome plans to more aggressively restrict fingerprinting across the web. One way in which we'll be doing this is reducing the ways in which browsers can be passively fingerprinted, so that we can detect and intervene against active fingerprinting efforts as they happen."

So again, it'll be real interesting to see what they do because there are so many little signals, like I was explaining. I just gave a handful of them. Panopticlick is the site where you can visit it, and many people have. And it does fingerprinting for you, and it shows you how many bits of entropy your browser has been essentially reduced down to. And what we would expect then is, when visiting the Panopticlick site with this next version of Chrome, that suddenly it's going to be far less sure than it has been in the past about who we are. So again, bravo.

And one more neat thing, that I hope other browsers are going to get, coming in Chrome is no more messing with the Back button. I'm sure all of us have entered a search term into our favorite search site and received a list of search results. We look them over using our prior experience of various domains that we recognize and then click on one that looks like maybe it's going to have what we're looking for. We browse around a little bit, maybe just look at the first site, and then press our browser's Back button. And nothing happens. Or maybe we go to a different page on the same site, not one that we were at before. In other words, the site we were at has been screwing around with the Back button.

I've had this happen at Microsoft's developer site pages due to the redirection system they built. Microsoft has created quite a tangle with their attempt at a browser-based centralized identification and authentication system where you get redirected and bounced around over to Live.com and then a few other places. And then the problem is redirects look like, to the browser, places you have visited. So when you hit Back from one of those, you don't go back to where you were before. You go nowhere. Typically, it bounces you forward again to the same page that you're on.

So that's annoying. And of course what I've taken to do, and what I imagine many of our savvy listeners have done, is you hold the Back button down, and after a little bit of an "are you sure" delay, you get then a dropdown list of all the URLs that are in this - essentially it's a most recently visited list, an MRU list, showing the URLs that it has in a stack. And typically what you'll do, for example, say that you were at Google, using Google search, you'll see several other weird-looking URLs and then finally Google's logo. And so I'll choose that one, and now I'm back to the search results.

Anyway, so there's a way around it. Sometimes, and I do this at Microsoft, if you hit the Back button several times quickly, you're able to step back before it's able to redirect you forward again, and so you're able to sort of hit back a couple times quickly to escape from that trap. Anyway, what Google announced is - and it's weird, too, because this has been a problem forever with browsers, ever since JavaScript was able to manipulate the stack. And that's what's happening is that JavaScript is able to push URLs onto the prior pages visited stack so that, when you hit the Back button, it takes you back to a page that the page you were on specified, rather than actually back to the page you feel that you were on before.

So again, not a hard problem to solve. Thank you, Google, for bringing this to us with Chrome. They're going to mark items on the stack which were visited as a result of previous user interaction, which is exactly what we mean when we hit the Back button. Which is to say, if I hit Back, I'm taken to the previous URL, the most recent previous

URL where I was actually doing something, where interaction took me away from it. So they're going to fix the Back button to prevent this from happening.

And in some of the editorializing I was reading about this, there was some hope that maybe Google, thanks to all the instrumentation that they have in Chrome, would actually notice those sites that were manipulating that stack to their users' annoyance, and maybe ding them in search results so that they're discouraged from messing with the previous pages visited stack. So we can hope for that.

Leo: This will do nothing to fix the problem of you must reload this form to continue, probably.

Steve: No, unfortunately. That's...

Leo: That's annoying, too.

Steve: I know. That's a very good point, Leo. That is annoying.

Leo: I understand why that's not really related, but...

Steve: Yeah. So as I mentioned at the top of the show, all Linux kernels before v5.0.8 are vulnerable. The good news is a very difficult to exploit, yet the bad news is a very potent, if it can be exploited, vulnerability that can be used for remote code execution. So this is known to the Linux community. A patch was released in late March and in mid-April. So it's a very complex problem. It was given - the NIST assigned it a high severity score of 8.1, yet a low exploitability index of 2.2. So it ended up netting out at an impact of 5.9 in this rating system that we have. So individuals have nothing to worry about.

But I would argue that cloud-based servers, because as we know, things that start out being difficult to exploit end up being figured out over time. And it's certainly going to be the case that there will be Linux systems on the Internet that are not going to be updated, that no one is paying attention to, that are just sitting there, running, doing their job, that somebody is going to be poking at. It takes advantage of what's known as the RDS, the Reliable Datagram Sockets protocol, which is transported over TCP. There's a use-after-free problem with it that can be exploited.

So I guess I would say to our listeners, if you are responsible for Linux servers that are Internet facing, you probably already know about the problem. You have probably already updated yourself. If not, probably do so. It's not a "run around hair on fire" kind of problem, but definitely worth doing. I expect that a few months from now we'll be talking about cryptocurrency miners being installed on systems that didn't get updated because the bad guys figured out how to do this. I mean, but it allows a 100% complete takeover of the server remotely. Again, I mean, it would be a real problem. It would make Heartbleed look like nothing by comparison, if it were easier to do. But as we've seen, these things only get easier over time. Which leads me into ZombieLoad.

Leo: Whoo.

Steve: Okay. So as I mentioned, I don't remember if it was before we began recording, Leo, or at the top of the show. But this just came out of embargo this morning. So I've got links in the show notes for anybody who wants to dig into it. I will cover it in some detail next week. I've been working on this podcast all morning, so I didn't have a chance to get into it. I have verified, however, that this day's, that is, today's Patch Tuesday, probably by no coincidence, has the updates for the firmware that Intel has been working on for a year.

Intel was informed of this more than a year ago, on March 18th, by the guys who first verified that this was possible. If you did not have updated firmware - and all versions of Windows 10 64-bit now as of today, today's Patch Tuesday, do have updated firmware on all processors where this is a problem. These are all machines made since 2011, so the last eight years' worth of Intel processors.

This is very reminiscent of Meltdown and Spectre, and it's considered to be positioned somewhere between the two. Less easy to do, that is to say more difficult, than Meltdown, but much easier than Spectre in terms of pulling it off. There is proof of concept available. The nature of what's happening is it's, again, it's a cross-core, or, I'm sorry, a cross-thread common core exploit. So, for example, turning off hyper-threading, where two threads are able to share the same physical core, that would be the only recourse if we didn't have now firmware available.

Intel has known about this, as I mentioned, for more than a year. So as this comes out of embargo, it's synchronous with available firmware across the board for all processors. I would imagine that, since Linux is able to install firmware at boot, that it will be updated. And as you mentioned, Leo, macOS has been updated. We now know that as of today Windows 10 is being updated. I imagine that Windows 7 users are probably left out to dry because Microsoft wants us all to be using Windows 10.

We'll see where this goes. But it's also the case that, if you have a motherboard from a reputable manufacturer, just as happened last year for the Spectre and Meltdown firmware updates, you could well see, like for example Dell is maintaining their systems to a high level of firmware release from Intel. So I wouldn't be at all surprised if Dell laptops and motherboards, and probably Lenovo also, do get some BIOS updates. That's the way we'll see them in systems where we're not able to get them updated through Windows 10 in Microsoft, is if our manufacturers support updating the BIOS. Then the BIOS is also able to patch Intel microcode on the fly.

So I'll know more next week. I don't know that there'll be much more to say. It is yet another microarchitectural flaw in the - really, for so long, as we've discussed, Intel was able to leverage lots of ways of speeding up the systems. But doing that left behind some state which affected other threads. And researchers coming along after the fact have been able to sense the state that was left behind by other threads. And in fact in the proof of concept, it's possible for one thread to determine previous websites that another user has visited, even though, for example, in a cloud setting you would hope to have much better virtual machine isolation.

This just breaks VM isolation in a way that is able to leak browser history, user-level secrets, passwords, keys, and system-level secrets, including disk encryption keys. So, I mean, it was - you could just imagine the poor Intel engineers who have their head in their hands, saying oh, my goodness, you know. Can we not have any processor performance increases without it causing a problem?

So anyway, more on that next week. I do have - no, I'm not going to go any further. That pretty much covers it. Because I had more written here in my show notes, but the fact that this has been patched for everybody who's on Windows 10 (64) means that it is not something that we need to be worrying about from this point forward.

I spent about four days last week adding support for Controlled Folder Access to my SQRL client for Windows. Controlled Folder Access is something that we mentioned back when it was introduced and announced by Microsoft in the now-infamous October 2018, what is it, 1709 Fall Creators edition. What was odd was that I had experimented with it and turned it on on one of my Windows 10 machines. And my own SQRL client was saying that there was no - that it couldn't find its SQRL identity on that machine. That is, the user's SQRL identity that I had previously loaded there. And so it had been on my list of things to look at before I declared that it was finished.

And as I think I mentioned before, I'm like right at that finish line at this point, where I've had a number of releases, about once a week for the last three weeks, that I thought that I was declaring as being the release candidate for the client. Anyway, finally I looked at it, and I realized that, if Microsoft ever got serious about Controlled Folder Access, then I'd have a problem that would confuse users. And that's what I wanted to avoid.

And we've seen what Microsoft does historically. Windows XP had a firewall, except there were lots of commercial firewall vendors at the release time of XP. And I don't know what Microsoft's thinking was, but it was there, but it was off by default. And our advice was always turn it on. Anyway, it wasn't until Service Pack 3 of XP that Microsoft finally turned it on by default. And by then it didn't have much impact because third-party firewalls had sort of gone away and weren't such a big deal anymore, or they figured out, I mean, they saw the handwriting on the wall, and they figured, well, we'd better go find out something else to do.

What Controlled Folder Access does is it is blanket prevention for any program that Microsoft hasn't blessed, prevention against modifying any of what is typical user content. So like everything in the My Documents folder tree - My Movies, My Photos, My Music. Basically they're billing it as protection against ransomware. As we know, what ransomware typically does is it wants to, and does if it can, encrypt all of the user's valuable content, their documents, the stuff that is not readily replaceable. The Windows core system, nobody cares about that because you can reload that. But you can't recreate your Ph.D. thesis if you're in the middle of working on it, and suddenly it's encrypted, and you don't have any current backups.

So Microsoft has this now in Windows 10 since 1709, the Fall Creators edition. And it will probably always be there because they generally don't take things away that they put in. It's off by default. And it turns out that it's not ready for primetime. When you turn it on, first of all, Microsoft says programs that Microsoft doesn't recognize will be blocked. Well, okay. So they didn't recognize my SQRL client. I get that. But they didn't recognize Internet Explorer, either, theirs.

So it turns out that, if you have it on much, you pretty much have to start whitelisting everything. Well, the good news is there's a whitelisting facility. And this thing's very noisy at the moment. Like doing anything, you get these little notifications sliding out from the lower right. And it says, "We just blocked something." And it's like, yeah, no kidding. Nothing works anymore. But if you then click on that, you're taken to a user interface page where you're able to ask for a list of recently blocked apps. And so it's easy to find what it was you were trying to do that they blocked and go, okay, yeah, well, that was me, on purpose. This wasn't ransomware. And so anyway, you start building up a whitelist of things that you're doing that it's blocking.

And I can understand how this being built in, many people will like it, will feel comfortable with it. They're happy adding things to the whitelist. And of course after a week you probably - you stop getting false positives because you've added everything that you're actually using to the whitelist. And if disaster befell you, and you did get hit with ransomware, it wouldn't be able to access any of the stuff in the areas of the system where you're probably creating content.

Anyway, so essentially what I did was I incorporated an awareness of Controlled Folder Access into the SQRL client so that, when it's installed in any system from 1709 later, it preemptively gives itself access, that is, it whitelists itself at that phase of installation where the user has elevated its privilege to allow itself to be installed. It also says, oh, while I'm here I might as well just put myself on the whitelist in case, whether Controlled Folder Access is enabled or not, so that at some future time either the user or Microsoft decides to enable it.

And this is my point about XP and the firewall lesson is that Microsoft generally sort of rolls these things out gradually. So it's there, but it's not on. They've got instrumentation watching everybody. And so maybe they're sucking in data. Maybe they're waiting for Windows Defender to get smarter about what things people are letting through. Who knows what's going on behind the scenes. But my point was I didn't want suddenly all SQRL users in the future to have their identities not seen. So anyway, that's where four days went last week was adding an awareness of that, figuring out how to do that. And that's in place now.

Now I return to working on the final document, which is what I had mentioned I'm now at work on, the SQRL Features, Capabilities, and Implementation document. My plan was to work up, to essentially put those all on matured web pages. But when I sat down to get to work, I thought, you know, I think I would just rather have this be in a document that people can download so that you can take it away, you can edit it, you can mark it up, you can do whatever you want. And it sort of starts at the beginning and successively gets increasingly more detailed all the way down to the protocol implementation so that managers can read the beginning of it and understand what the features and capabilities are, and people wanting to implement it can start with that, but then go into much detail. So once that's finished, the project is finished.

I will be giving a second presentation to a "local to me" group, and I've asked them for permission to mention it on the podcast. I don't know how many people who listen who are in the Los Angeles area might be interested in attending. But it is the Open Web Application Security Project, L.A. Chapter. It's Wednesday after next, which is May 22nd, at the OWASP L.A. Monthly Dinner Meeting. And so if you google OWASP L.A. Monthly Dinner Meeting, they organize through Meetup.com, and you could say that you want to be there, register yourself, and say hi to me as a Security Now! listener. And I have another invitation pending at OWASP in Dublin. I am in the process of getting my passport renewed so that I will be able to accept that invitation.

Leo: Oh, good. That's exciting.

Steve: Yeah, those are fun.

Leo: How long has it been since you've been out of the country?

Steve: Leo, not since I was with you in Toronto and Vancouver.

Leo: Oh, yeah. I guess that's out of the country.

Steve: That's why I got my passport, when you said, "Hey, you know, you want to come and do some shows with me in Toronto?" And I said, "Oh, I'm going to need a passport."

Leo: Wow. So I'm thinking Lorrie's going to take advantage of this, and you'll be going to Paris soon.

Steve: Well...

Leo: You better.

Steve: We'll see. If somebody in Paris wants a SQRL presentation...

Leo: Oh, come on. You can take her to Paris. You don't have to have a business reason.

Steve: No, I've got work to do on SpinRite. I've got a bunch of people who are saying get back to SpinRite.

Leo: Oh, okay. Good, good, good. No, that's good. Yup, that's good.

Steve: Anyway, and I did want to also mention that blog.grc.com, where I have been, did expire. I had the domain-forwarding option, and so I posted yesterday that I had 24 hours left. Anybody who now goes to blog.grc.com will come to the new blog which I've put up. And I plan to explain, to lay out my roadmap for SpinRite, since it's beginning to take up a little space in my brain once again.

Leo: Yay.

Steve: Since SQRL is pretty much behind us.

Leo: Oh, hallelujah.

Steve: Yeah.

Leo: I speak for everyone when we say 6.1, 6.1.

Steve: Getting back to it. Yup, yup. Yes, indeed.

Leo: Yes, indeed.

Steve: So, okay. A bunch of stuff to talk about with Android Q. As I mentioned before, I was very impressed with the things that Google has been doing and the investment that they're making in this. And so there are several different aspects. One, as you mentioned, was Project Mainline. Stephanie Cuthbertson, who's the Senior Director for Android, explained last week during Google I/O, she said: "Your regular device gets

regular security updates already, but you still have to wait for the release, and you have to reboot when they come. We want you to get these faster. Even faster." And she said, "And that's why in Android Q we're making a set of OS modules updateable directly over the air, so now these can be updated individually as soon as they're available and without a reboot of the device."

And of course what that says is that they're doing something more like the model that we've now gotten used to with Chrome, where it's just sort of a rolling latest release, and it's being updated constantly. Essentially that's what this is known as. It's known internally as Project Mainline. Google developers have spent the last year working to split 14 OS core components into separate modules. So even though they're core OS modules, they will be behaving more like applications, which are able to be updated on the fly. Google has said that it's going to be pushing the update to all devices which support the mechanism.

Essentially what it'll do is it'll be able to stop that subcomponent of the OS, to halt it, update it, and then restart it without essentially any impact on what the user is doing. So no need to reset and reboot the OS. It'll just do it on the fly. So the modules are ANGLE, APK, the Captive Portal Login, Conscrypt, the DNS Resolver, the Documents UI, External Services, the Media Codecs, the Media Framework - and of course we know the media platform has been a big source of problems in the past, so it's very cool that those can be on-the-fly updated - Network Permission Configuration, Network Components. And once again those are Internet-facing, so they tend to be problematical.

So again, if you wanted to choose things that you'd be able to update on the fly, that's where it would be. Also the Permission Controller, Time Zone Data, and Module Metadata. So those are all internal core services. The user doesn't directly see those, so they're not like apps. But they are often the components where the most security problems can be found. So again, this was not an easy piece of work.

What's interesting is that - and I wished it was going to be more available. The Verge, who did some reporting on this, learned that as individual device makers will be able to opt out of the new feature, although I don't know why they would choose to, just seems like a big bonus and plus for their users. But individual device makers will be able to say no, we don't want to do that. Also it turns out that only devices which initially ship with Android Q will be able to have this functionality.

So although older devices will be able to be upgraded to using Q, they won't be able to take advantage of Project Mainline, I guess because it's just very tricky to do, and maybe there's some hardware-side support that's necessary. Or maybe it's from a security standpoint. It could be that, in order to prevent any malicious abuse of this technology, Google has had to incorporate some things from the start that prevent it from being tampered with. I'm just guessing there. But for whatever reason, unfortunately, devices that aren't purchased with Android Q will never, although you'll be able to get all the benefits of Android Q, you won't get this particular on-the-fly updating of these components. So anyway, still a nice step forward to Q from Pie, which is where we've been.

Also they announced additional encryption benefits. They explained that storage encryption is one of the most fundamental and effective security technologies, as we know, but that current encryption standards require devices that have cryptographic acceleration hardware. Because of this requirement, many devices are not capable of using storage encryption. But remember that we talked last year with their innovation of Adiantum, which was their very cool storage encryption technology which, unlike AES, which is the technology that is often used - the problem with AES is that its pure software implementation is necessarily slow because the bit twiddling that has to be done

to implement the AES cipher requires many instructions, many generic instructions in order to pull off the bit twiddling that AES requires.

That's why Intel has the so-called AES-NI instruction enhancements, "NI" standing for New Instructions. And essentially those move into firmware in order to speed them up, the bit twiddling that the AES cipher requires. So there is a very fast stream cipher which can be implemented in software, known as ChaCha20. It only uses the simple basic instructions that are fast on all processors. But the challenge there was to come up with a cipher which would not expand the size of the block. Remember we talked about this before. It's fine in a stream cipher, like over TLS, to have to add a little bit of padding or an initialization vector to the cipher. In a communications protocol, you don't care if it makes it a little bit longer. But you can't do that on a block storage protocol because there is nowhere extra to store any additional stuff.

So Adiantum is Google's solution for very fast, non-size-increasing encrypted content. And we get that with Android Q. In fact, all Android Q devices are going to be required to support full storage encryption because it'll be built into Q, and it's free. It no longer costs anything in terms of performance, even if you are on a non-AES-NI-compatible or capable platform that would otherwise normally make it much smaller.

They said: "Our commitment to the importance of encryption continues with the Android Q release. All compatible Android devices newly launched with Android Q are required to encrypt user data, with no exceptions. This includes phones, tablets, televisions, and automotive devices. This will ensure the next generation of devices are more secure than their predecessors, and allow the next," as Google said, "the next billion people coming online for the first time to do so safely."

They also said: "However, storage encryption is just one half of the picture, which is why we are also enabling TLS 1.3 support by default in Android Q." Of course we've talked about that. It makes handshakes quicker. It supports a next generation of later, more secure ciphers. It explicitly drops support from the older SHA-1 hash-supporting cipher suites and so forth. So that'll also be built into Android Q. So another very nice bump forward in security.

Under platform hardening, this is where I was most impressed. I first have sort of an overview of that. Then I'm going to dip into it a little bit further. They said: "Android utilizes a strategy of defense-in-depth to ensure that individual implementation bugs are insufficient for bypassing our security systems." Again, individual implementation bugs are insufficient for bypassing our security systems. So the idea being that they've recognized that, despite a history of attempts to write secure software, stuff still gets through. So the solution is to acknowledge the reality that stuff is going to get through and to make the consequences of stuff getting through much less severe.

So they said: "We apply process isolation, attack surface reduction, architectural decomposition, and exploit mitigations to render vulnerabilities more difficult or impossible to exploit, and to increase the number of vulnerabilities needed by an attacker to achieve these goals." And I just love the phrasing of that. It's a sober and realistic expression of the truth that we have seen of the challenges facing any highly targeted platform, and Android is arguably the number one most targeted platform now that exists.

They said: "In Android Q we have applied these strategies to security-critical areas such as media, Bluetooth, and the kernel. We describe these improvements more extensively," they said, "in a separate blog post," which is where I'm going to go next. But the highlights include a constrained sandbox for software codecs, which is what we'll be focusing on in a second; increased production use of sanitizers to mitigate entire classes of vulnerabilities in components that process untrusted content.

Now, actually I've referred to this both in my own code and in general. The idea is that you perform sanity checking on parameters. When you have something that is parameterized, you say, rather than just assuming it's correct, you clamp it with a sanity check to say, wait a minute, does this make sense? Can a bitmap image actually be this big? I mean, is it sane? And so you can catch a huge class of problems just by applying what they call "sanitizers" and I call "sanity checking."

They said also a shadow call stack which provides backward-edge control flow integrity. Control Flow Integrity (CFI) is a mitigation that we've talked about also in the past. Also protecting address space layout randomization against leaks using execute-only memory, which is again another means of hardening. And just also tightening up heap-related vulnerabilities, making them much more difficult to exploit. They explain under authentication that Android Pie, as we know, introduced the BiometricPrompt API to help apps utilize biometrics including face, fingerprint, and iris. They said since launch they've seen a lot of apps embrace the new API.

Now with Android Q they said: "We've updated the underlying framework with robust support for face and fingerprint. Additionally, we expanded the API to support additional use cases, including both implicit and explicit authentication. In the explicit flow, the user must perform an action to proceed, such as to tap their finger to the fingerprint sensor." Or, if they're using face or iris to authenticate, for example, where they might already have their face in view, the user must still explicitly click an action button to acknowledge, to proceed.

They said: "The explicit flow is the default flow and should be used for all high-value transactions such as payments." And, for example, on the Android client for SQRL, the user is being asked to verify the identity of the site they're wanting to authenticate to and then acknowledge that explicitly - which is just a touch of a button on the screen in order to say yes - because, if we use the implicit flow, that would happen just instantaneously. I mean, SQRL is instantaneous, virtually. But in this instance we want to make sure that the user knows, has acknowledged that they're wanting to use their identity to authenticate to the site. And so we want to just say, look, just make sure this is where you want to be. And then you tap the button, and off they go.

They said: "The implicit flow does not require an additional user action. It is used to provide a lighter weight, more seamless experience for transactions that are readily and easily reversible." So you can imagine if you have something that just doesn't matter, if the camera sees you, recognizes who you are, just allow it to happen without any intervention. So that would make it additionally, for those sorts of things, really very smooth.

Then they finished up on that, saying: "Another handy new feature in BiometricPrompt is the ability to check if a device supports biometric authentication prior to invoking BiometricPrompt. This is useful when the app wants to show an 'enable biometric sign-in' or similar item in their sign-in page or in-app settings menu." In other words, apparently until now there hasn't been a way for the app to determine whether that's available or not. "So to support this," they said, "we've added a new BiometricManager class. You can now call the canAuthenticate() method in that class to determine whether the device supports biometric authentication and whether the user is enrolled."

So again, very cool stuff. And they did say - they gave us a little bit of a tease. They said beyond Android Q they are considering, they're looking into an Electronic ID support for mobile apps, so that your phone can be used as an ID, such as a driver's license. They said: "Apps such as these have a lot of security requirements" - tell me about it, because that's exactly what I've implemented with SQRL - "and involves integration between the client application on the holder's mobile phone, a reader/verifier device, and an issuing authority backend system used for license issuance, updates, and revocation." And of

course, as our listeners know, that's what I explicitly avoided needing to have in the SQRL solution. It's a two-party solution, not a three-party solution.

Anyway, they said: "This initiative requires expertise around cryptography and standardization from the ISO and is being led by the Android Security and Privacy team. We'll be providing APIs and a reference implementation of HALs [Hardware Abstraction Layers] for Android devices in order to ensure the platform provides the building blocks for similar security and privacy-sensitive applications. You can expect to hear more updates from us on Electronic ID support in the near future." And so Leo, what I still want to know, the outstanding question.

Leo: Yes.

Steve: What dessert is Q going to be?

Leo: No one knows. There is nothing it could be, yeah.

Steve: Anyway, the last thing I wanted to touch on was the specifics of the security enhancements. I've got a pie chart here in the show notes which shows from their own metrics a really interesting pie chart, the vulnerabilities by component. And it won't surprise any of our longtime listeners that by far the largest source of vulnerability, this big huge blue chunk of the chart, which is easily a third of the whole pie, is media.

We know what a problem media is. The codecs, the so-called compressors/decompressors, they are interpreters because media has become so complicated that essentially you are running an interpreter to read metadata and metatags in the media which then describes the content which follows, little blocks of content which follows, audio chunks and video chunks and so forth. And so you're running interpreters which are just - it's the nature of developers to assume that the content they're receiving was authentically created by a compressor when they're being asked to decompress it. It's just difficult not to think that way. So fully a third on that pie chart is media.

The next biggest thing, not quite another third, but still large, is Bluetooth. Number three is the kernel. Number four is NFC, near-field communications. Then the framework, the system, graphics, miscellaneous things. Then we're getting down into little tiny pieces of the pie: camera, WiFi, and other. So Android Q, which is now available to developers in beta, has continued to tighten up its code and to harden its various attack surfaces. And by examining the problems that they have found, they were, well, and rather than thinking, oh, well, that's a one-off that will never happen again, again I applaud the maturity of their approach. They said, okay, we're having problems in this particular area of Android. Let's assume we're going to have more problems in the future. What can we do to fix it? So number one problem was media. Number two was Bluetooth.

So as their engineers explained, they said: "In Android Q, we moved software codecs out of the main MediaCodec service into a constrained sandbox." They wrote: "This is a big step forward in our effort to improve security by isolating various media components into less privileged sandboxes." And they quote a blogger, Mark Brand. They said: "As Mark Brand of Project Zero points out in his 'Return to libstagefright' blog post, constrained sandboxes are not where an attacker wants to end up." Meaning breaking out of a codec you want to get into the kernel, which until now is where you've been able to get to. No longer.

They said: "In 2018, approximately 80% of the highly critical, high severity vulnerabilities in media components occurred in software codecs, meaning further isolating them is a big improvement. Due to the increased protection provided by the new mediaswcodec sandbox, these same vulnerabilities will receive a lower severity based on Android's severity guidelines." Meaning, again, if there's a problem, they're not going to hurt users so much.

So I mentioned before that this was an iterative process, that is, over time. So we have before N. Then we have N, O, P, and Q. Prior to N, these media services were all inside a single monolithic media server process, and the extractors were run inside the client. And of course this is where problems like Stagefright were able to cause such problems so that, for example, just as we recall, sending someone a text message was able to compromise their device.

Then in N they explained that: "We delivered a major security re-architect, where a number of low-level media services were spun off into individual service processes with reduced-privilege sandboxes. Extractors were moved into the server side" - which is to say out of the client - "and put into their own constrained sandbox. Only a couple of higher level functionalities remained in Mediaserver itself.

"Then in O and P, the next generation of Android, these services were 'treblized'" - they turned it into a verb, treblized - "and further deprivileged, separated into individual sandboxes and converted into HALs," into Hardware Abstraction Layers. Essentially, and we talked about this at the time also, they moved this away from the third parties. Previously those had been third-party components. And so by pulling them into hardware abstraction layers, they were able to isolate what needed to be per-hardware implementation and essentially move more of it under their own control. They said: "The media.codec service became a HAL, while still hosting both software and hardware codec implementations."

And now finally in Q, which is, as I mentioned, at beta, not yet in people's hands, but it's on the way, the software codecs were extracted from the media.codec process and moved back to the system side. So again, they're continuing to take more responsibility. But this is where we want to have it because they're also now able to add on-the-fly updates to these things. They said: "It becomes a system service that exposes the codec HAL interface. SELinux policy and seccomp filters are further tightened up for this process. In particular, while the previous media.codec process had access to device drivers for hardware accelerated codecs, the software codec process has no access to device drivers."

So again, by continuously working to see where their problems are, and incrementally changing the hardware architecture, they're able to redesign the system to dramatically enhance the security. And I'm just - I'm very impressed by the maturity of the development philosophy that I see in those details. Google is clearly very focused upon carefully studying the sources of problems, applying sound engineering solutions, inventing new solutions when they need to. And they're just patiently evolving an increasingly more stable and capable software operating system platform. You know I've given them a lot of heat in the past over the problems that they've had on the security side. But I'm impressed by what I'm seeing. I really believe that Android is shaping up to be an asset of tremendous value for the industry.

Leo: I'm glad to hear it, since that's pretty much what I carry around with me all the time.

Steve: Yup, yup. It's becoming more and more secure. The ability to have always present whole device encryption, I mean, on any hardware, on any hardware that has Q they'll be able to have whole device encryption. And the fact that they're now able to - and you can see how by pulling pieces back under their own control, they're then able to enhance it with on-the-fly encryption, meaning that the manufacturer only needs to deal with a much smaller aspect of the hardware interface to Android, and Google is able to then be the caretaker, and a responsible caretaker, for a much larger piece of this. You know, it's increasingly becoming the Apple iOS model, where in this case Apple has the whole pie. Google is increasing their share, but also being a really very responsible caretaker for it.

Leo: Nice.

Steve: So bravo, yup.

Leo: I just got my April Update on my Samsung Galaxy S10 Plus, so it's all working out.

Steve: Yay.

Leo: Steve Gibson is at GRC.com. You should be at GRC.com. Go there. Find out what's going on with SQRL, its imminent release. SpinRite, of course, his bread and butter, the world's best hard drive maintenance and recovery utility; Shields UP!; all the great stuff he does there. And this show is there. You can download 16Kb versions for the bandwidth impaired, full 64Kb versions for those who like to spend their bandwidth with abandon. Also handwritten transcripts, which I suppose is the lowest bandwidth version of the show.

Steve: Yes.

Leo: Probably the smallest file size, thanks to Elaine Farris.

Steve: And even compresses way down, too.

Leo: Yes, yes. If that's what you care about. If you're on a boat, and you're using satellite Internet. We have audio and video of the show at our site, TWiT.tv/sn. And you can of course always subscribe. And that's probably the best thing to do. If you want to watch us do it live, TWiT.tv/live. That's where the streams live, audio and video. And we do the show normally about 1:30 Pacific, 4:30 Eastern, 20:30 UTC of a Tuesday afternoon. If you do come by live, join us in the chatroom at irc.twit.tv. That's where the cool kids hang out. Steve, have a great week.

Steve: Thank you, my friend. We'll see you for 715.

Leo: Hallelujah. See you then.

Steve: Next Tuesday. Bye.