WebAssembly

Description: This week we discuss Intel's Spectre and Meltdown microcode update, this week in cryptojacking, Tavis strikes again, Georgia on my mind (and not in a good way), news from the iPhone hackers at Cellebrite, and Apple moving its Chinese customer data. ePassports? Not really. Firefox 60 loses a feature; the IRS and cryptocurrencies; Android P enhances Privacy; malicious code signing news; a VERY cool CloudFront/Troy Hunt hack; a bit of errata, miscellany, and closing-the-loop feedback from our terrific listeners; and a closer look at WebAssembly.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-652.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-652-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Lots to talk about. Tavis Ormandy's found another big flaw, this time in uTorrent, the BitTorrent client that a lot of people use. You may not want to use it after you hear about this. News from the iPhone hackers at Cellebrite. Apple's moving its databases to China. And Firefox 60 is losing its cookie management feature. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 652, recorded Tuesday, February 27th, 2018: WebAssembly.

It's time for Security Now!, the show where we cover your security and privacy online with the King, Steven Gibson of GRC.com, the King of Security. Actually, you...

Steve Gibson: I'm not sure who crowned me, Leo, but...

Leo: Yeah, you wouldn't want that title, I don't think.

Steve: No. And besides, when you take the crown off, it leaves this line across your forehead. It's really annoying.

Leo: I hate it when that happens.

Steve: You just don't want that.

Leo: I hate it. Last show of February 2018.

Steve: It is indeed. I titled this "WebAssembly," not because I want to do a super extensive deep dive, but because a whole bunch of our listeners have asked me about it since I've been talking about the inefficiency of mining cryptocurrencies with JavaScript. They're all saying, well, what about WebAssembly? What about WebAssembly? So I thought, okay, let's talk about that, sort of put that in the proper context, like where that fits in terms of performance and so forth. So mostly we're just going to kind of wrap up with a, like, what is WebAssembly - what are the benefits, where does it stand, where does it fit in the performance range and so forth. So I thought, okay, that's what we'll call this February 27th Issue 652 of the podcast.

But other than that, we had a bunch of interesting news. We do have an update from Intel on the status of Spectre and Meltdown microcode. And there's a link in the notes, Leo, to a PDF you're going to want to open at some point, just probably for your own edification. I think our listeners will, too, because it's the complete processor breakdown - which is probably not the right word in this case - of where Intel stands with every single processor they have across their line in terms of the microcode updates. And so we're going to talk about that.

Also we have this week in cryptojacking, some news there. Tavis strikes again. Georgia on my mind, and not in a good way. News from the iPhone hackers at Cellebrite, or at least rumors. Also some concerns over Apple's announcement that it's going to be moving its Chinese customer data to China. A question about ePassports, which were mandated back in 2007, and something you're not going to believe. Also Firefox 60, which is the next major release to come out, is losing a feature that I expect some of our listeners are going to be unhappy about. It kind of annoys me, but progress. We also have some news about the IRS and cryptocurrencies. Some good news about the next release of Android, which will be the "P." And in some ways it looks like it enhances privacy, so maybe "P" is for Privacy, also I guess it has to be, what, Pancakes or something.

Then we also have some updates on malicious code signing, a very cool CloudFront/Troy Hunt combo hack which I think is just very, very clever. Some errata, a little bit of miscellany, some closing the loop with our terrific listeners, and then we will wrap up by explaining where WebAssembly fits in the spectrum of browser-based code execution because it's not the answer for cryptocurrency mining, but it's definitely an answer to something. So we'll figure that out.

Leo: We'll figure out what the question is in a bit.

Steve: And, oh, another Picture of the Week, Leo. This is just - oh, lord.

Leo: That cracks me up.

Steve: Yeah.

Leo: It's got to be a joke; but, well, maybe not.

Steve: I think it also - I really think it has to be. I mean, it looks completely legitimate. But, you know, it may be a hoax. Although I think the one with the pail, remember the generator...

Leo: Yes, this is a series, an ongoing series.

Steve: Yeah. If nothing else it's good for a hoot. You know, I've been staring at this Picture of the Week while you were telling us about WordPress. And I think I understand it better than I did before.

Leo: Well, I think anybody looking at it's going to get it pretty quickly, at least what the intent is; right?

Steve: Yes. Well, and of course the first one we ran across like this was that big motor generator that was near some railing. And there was a big green solid copper, but green insulation for ground, wire coming off, wrapped around a pole, a metal pole, that was then stuck into a big pitcher, a bucket of dirt. And of course we all had a lot of fun with that because some...

Leo: That one looked like somebody believed it would work.

Steve: Yes. It looked like they actually thought, okay, I'm grounding this generator, you know, into [crosstalk].

Leo: And we should mention, it might even be easier to understand if you used the British term for it, which is "earth."

Steve: Yes.

Leo: Right?

Steve: Yes.

Leo: We're sending it to earth.

Steve: Yes. And sometimes we use both, an "earth ground."

Leo: Yeah.

Steve: Okay. So just for our listeners who can't see this, here we have kind of off of the image is some big high-power switchbox thing, and we're just seeing the lower right-hand side of it. But then as code requires, apparently, coming out from a nice grommeted connector is this yellow-and-green-striped wire that is clearly meant to be ground. It goes into some other thing that probably some other code requires. And then out it comes again through an anchoring post and then into a rather small-sized plastic bag...

Leo: It's a one-quart Ziploc bag with dirt in it.

Steve: Yeah, a sandwich would not really fit in this thing. And it's got some dirt in it. And then in case somebody wasn't immediately...

Leo: Now, why is that dirt hanging from the power box?

Steve: Exactly. There's also a round yellow sticker with the famous "ground" symbol on it, the line coming down with three horizontals like in a triangle.

Leo: This is our ground. This is our ground, yeah.

Steve: So you clearly know. So at first I was thinking, okay, this looks just like, you know, like, really? But I realized then that that backing that we're seeing looks like it's metal. It's all, like, sheet aluminum because we can see a rivet over on the right-hand side. So, and then if you look at the wire coming out, there's a post in the steel or this aluminum or whatever it is...

Leo: That's the real ground; right?

Steve: Yes. I think that's the point is this entire back sheet is itself properly grounded. And so the wire comes out and just attaches to a post, an electrically conductive post. And then somebody was having some fun. And so they said, okay. Or maybe they wanted to make it clear to somebody looking at it that the whole back sheet was a ground, so they came up with this little kind of cartoonish thing just to sort of say, look, yes, you can't tell, but the whole backing sheet is hooked to earth, and so we're just going to hang a little bag of dirt here and connect it up just to drive that point home visually. Anyway, a fun picture.

Leo: Yeah.

Steve: Okay. I talked about this PDF. I've got the link in the show notes. I think if you were to Google "microcode update guidance," that's from the URL of the

newsroom.intel.com, where our friend there is still referring to himself in the first person. Anyway, I think that will probably take you to the update.

So this was released last week with the latest news of where Intel is in this slog that they're in, this trek to produce new microcode, like across their entire product family. The announcement that goes along with this PDF states that Intel has now released production microcode updates to their OEM customers and partners for Kaby Lake and Coffee Lake-based platforms, plus additional Skylake-based platforms. This represents their sixth, seventh, and eighth-generation Intel Core product lines, as well as their latest Intel Core X-series processor family. It also includes their recently announced Xeon Scalable and Xeon D processors.

Now, the problem is Intel's got just a ton of processors. And so this PDF does allow someone to go in and figure out what the heck, you know, like is theirs there yet? Now, this doesn't mean these are available to end users, and I noted in some of the coverage on the 'Net that Intel had not yet updated that Linux package which would allow Linux systems to dynamically patch their microcode. And I don't remember, Leo, a couple weeks ago I ran across some news that Windows had the ability to do microcode patching. And I don't remember if I ever mentioned it on the podcast. There are two DLLs in the Windows System32 directory whose names make it very clear they are for Intel and AMD microcode patching. And so this suggests that, if Microsoft wished to, they could perhaps do this, although that seems like more responsibility than they're going to want to take.

Anyway, so what I was going to say was that this is the release of this microcode to Intel's OEM customers like Lenovo and Dell and everybody else, not to end users. Linux end users can potentially get this as soon as Intel updates the Linux package, but the rest of us are going to have to wait. But at least this, for anyone who's interested or for whom this may be mission critical, this indicates where - this updated PDF indicates where Intel is in the process. And, what, maybe a third, maybe a quarter of the listings there, there was lots of beta and pre-beta and in-planning, which means not yet in production. The label "production" in that one column is the only one that says yes, we've pushed this out the door, and we're not going to pull it back again, hopefully. I'm sure they learned their lesson from the first round.

So anyway, they're moving forward. And as soon as the OEMs are able to verify that it works and test it, I think what we'll see then is another round of BIOS updates which we as end users will then be able to download and run on our laptops and desktops in order to get this Spectre Variant 2 is the one, is kind of the nastiest one that's been causing all the trouble. Meltdown, that got fixed pretty easily. The Variant 1 of Spectre was not a big problem. This is the really nasty one is Variant 2, which we're now still sort of waiting for final fixes for.

I thought, okay, I'm going to call this This Week in Cryptojacking. And by the way, I thought I had coined a fun term last week, mal-mining. That's still not bad.

Leo: No, I like it, yeah.

Steve: Yeah, mal-mining. Except that cryptojacking, I guess, has already become the official term. So I was late to the party with mal-mining, although it makes it a little more clear, I think. I like mine a little bit better. Anyway, a group called RedLock Cloud Security Intelligence, or RedLock CSI, they discovered that Tesla, our Tesla Corporation, had become the latest victim of cryptojacking. They started their report by saying: "We

are beginning to witness the evolution of cryptojacking as hackers recognize the massive upside of these attacks and begin to explore new variations to evade detection."

Okay. So what happened is, in the case of Tesla, this wasn't some massive corporate-wide infiltration. But there was something misconfigured, as so often is the case. They discovered that one of Tesla's Kubernetes consoles, believe it or not, was not password protected. It was just wide open. Kubernetes, I know I've heard you use the term before, Leo. For those of our listeners who don't know, it's an open source platform originally developed by Google which has now then been released and is independently maintained and further developed by the Cloud Native Computing Foundation. And essentially it's an automation system for Linux containers which eliminates a lot of the manual processes involved in deploying and scaling containerized Linux apps, so sort of a meta container management console, and you need to give it a password. Apparently there was an unsecured Kubernetes "pod," as it's called, that was Tesla's. By getting into that, and I saw a screenshot of this as I was digging around, there just plain as day were the two Amazon S3, the Simple Storage Service, the ID and the key.

Leo: Oh, no. Just like in a picture?

Steve: Just right there.

Leo: Oh. Oh.

Steve: Right on the console welcome screen.

Leo: Of course that's the secret key. If you give it out, you have access to those buckets; right?

Steve: Yes, yes. That is, you know, that's what you have to absolutely protect.

Leo: That's your password, basically.

Steve: Yes. Yes, it is.

Leo: Oh, criminy.

Steve: So there was exposure of sensitive data. And I saw a reference to telemetry, but it wasn't clear what telemetry, and it really isn't important. What was interesting was that, in addition to that data exposure, as a consequence of having access to this console, the hackers were performing some, well, they were able to run crypto mining, but of a slightly different nature, or sort of more professional than we've seen before. They were using a number of advanced evasion techniques in order to keep from being discovered. It didn't keep them from being discovered ultimately.

But it's interesting that we're seeing this sort of next-generation evolution. For example,

they weren't just running something from Coinhive.com. They did not use a well-known public mining pool, which is like the easy thing to do. You just run some mining pool software and point to a public mining pool. The problem is then you have well-known, well-identified endpoint for your traffic, and it's easy to catch.

Instead, they installed their own mining pool software and then configured the malicious script to connect to an unlisted semipublic endpoint. They even used an IP address for the mining pool software behind Cloudflare because, whenever you register a CDN, you get an IP. And this allowed them to use an IP that could be changed at will and was floating around, so not locked down. They also went to the effort of using a nonstandard port. So they were off the radar. And the RedLock people who saw this all running noted that the CPU was not near pinned.

So these guys, whoever the bad guys were who set this up, they went to some lengths to install something which would be able to run probably undetected for a long time. They basically created their own command-and-control infrastructure so that it would not be easily found from scanning or typical behavior, and then didn't want to give themselves away by going full bore, cranking up the CPU usage, because that could draw attention to them. So they were taking some percentage of the CPU for mining, but probably hoped to just sit there and mine quietly for the duration of however long they could get.

So I think that the take these guys had, the RedLock folks had, is exactly right, that what we're going to begin seeing, what we're going to be seeing is an evolution in the seriousness of cryptojacking or mal-mining, to use my term, that is serious. Through the years, viruses never made anyone any money. They were just sort of curiosities. They were annoying. They propagated. They were sort of in the domain of hackers who you sort of wondered, well, why are they doing this? What's the point? It was just sort of, oh, look, you know, I can screw around with the world's computers by creating a virus and seeing how far it goes or how long it lives or something. But they weren't money-making.

But then, as we saw a couple years ago, when this idea of cryptography-based ransomware occurred, where you could arrange to create a key where each instance's key was unique, and you were able to use public key crypto in order to encrypt someone's files and then hold them at ransom, we saw a change then. Suddenly there was a way to make money, potentially, from these kinds of, I mean, the same kinds of viruses and games that were being played before. Suddenly dollars. And of course now we have real world currency exchangeable cryptocurrencies. So this is not just virtual money, this is money you can buy things with.

So now what we're seeing is the next evolution of this, is that now it's possible to monetize the theft of idle CPU resources, which arguably exist everywhere. I mean, you know, everything that's on the 'Net has a computer behind it. And suddenly, I mean, and sadly, light bulbs are, I mean, all kinds of stuff, all of our IoT stuff may not have much power, but there's strength in numbers. So I really think we're going to begin to see, well, we are seeing, and I think it's going to continue, I see no sense that it's going to reverse. I remember when we first saw the tip of the ransomware iceberg years ago. On this podcast we said, uh-oh, this is not good. Now you can get paid for infecting and encrypting someone's computer. That creates incentive.

And I think the same pressure, the same leveraging of our unfortunately porous security is going to create I guess what we would call "intrusion pressure" to get into systems exactly like this, find something, and just begin setting up little mining operations wherever you can and participate in a pool and start generating revenue. In the same way that the first sign of the ransomware sort of gave us this sense that, uh-oh, that we're going to be seeing more of this, and of course we did, I think we're entering a new

world where the ability to monetize idle CPU resources, like other people's - because remember, depending upon where you are and what technology you're using, there's so much competition for mining now that the cost of electric power is oftentimes greater than your ability to mine a given cryptocurrency.

That is, your electric bill goes up more than cryptocurrency comes into a wallet. Again, depending upon where you are, because there's huge variation, at least in the U.S., in the cost of electricity. And depending upon the size of your installation, a percentage of that electrical power is not just going to computation, it's going to heat, where there's a lot of heat generated. So now you have to somehow get the heat out of the environment, so you have to pay for air conditioning, too, you know, classic datacenter operations.

So the point is you don't want to use your own power. It's not profitable. You want to steal power from other people, no matter how inefficient it is, because you're not paying for it. They are. So anyway, I think this is going to be, like 2018 is going to be the year of the rise of this intrusion pressure, that is, we've talked about how porous security is, that if somebody really wants to get in, they can push and push and scan and probe and phish and use social engineering, I mean, the more you want to get in, the greater the probability is you can. And the idea that there's a profit, potentially long-term persistent profit - think about, for example, the fact that the Internet is all full of this background radiation that we've talked about, that Code Red and Nimda worms are still out there scanning around. Well, they could be mining cryptocurrency for the last decade. Well, it hasn't been around for the last decade.

But the point is there is an opportunity to set up long-term residence, stealing power from somebody's closet. Who knows where these things are? Anyway, I think This Week in Cryptojacking - or mal-mining, I just have to settle on a term one of these days - that's going to be an ongoing theme, I think, for the podcast.

Tavis, our illustrious Google Project Zero security researcher, peered into uTorrent, and we can pretty much predict the rest. As we know, uTorrent - is it pronounced "u-torrent" or "micro-torrent"?

Leo: I say "micro-torrent." I think the "u" is supposed to be the micron symbol; right?

Steve: Yeah. That's always sort of been [crosstalk].

Leo: Yeah. I always say "micro-torrent."

Steve: Okay, good. And of course that is BitTorrent's official peer-to-peer file torrenting application which is in use, I mean, by many millions of active users every day. I mean, it's the Internet's No. 1 peer-to-peer torrent application.

After checking into the way uTorrent operated last November, Tavis, on behalf of Google's Project Zero, not surprisingly found some problems, actually some bad problems, and reached out to BitTorrent. It was another instance maybe of him not knowing who to contact, or them not knowing who he was, or being busy, or who knows what is going on at BitTorrent. But he didn't get any dialogue with them, despite the fact that what he found was a relatively easily exploitable remote execution vulnerability in uTorrent, which, whoops. That's not good.

So finally, as the clock's ticking - remember that Google's Project Zero gives companies 90 days to repair from the time they are notified, feeling that that should be enough time for anyone to respond with a fix to a security vulnerability, yet still protect users by not making it two years. Make it three months.

So last week Tavis sent another tweet to BitTorrent's original creator, Bram Cohen, who's now doing cryptocurrency stuff. He tweeted: "I don't think BitTorrent are going to make a 90-day disclosure deadline. Do you have any direct contacts who could help? I'm not convinced they understand the severity or urgency." Well, it turns out that BitTorrent had been listening, just hadn't been corresponding. A "fix," and I've got that in quotes in my notes because they didn't fix it, for the problem is currently in the beta channel, but not yet in a stable release.

Well, Tavis is not impressed, and he wrote in the Project Zero blog, he said, "Hmm. It looks like BitTorrent just added a second token to uTorrent web. That does not solve the DNS rebinding issue." That's what it was that Tavis found was a vulnerability that could be exploited through DNS rebinding. He said: "It just broke my exploit," meaning that unfortunately they prevented the one thing that Tavis found, or the one example Tavis gave them, but didn't fix the underlying problem. They muted the symptom, essentially, rather than curing the disease. And in this case the disease appears to be significant.

What Tavis wrote in the blog was, he said: "By default, uTorrent creates an HTTP RPC [Remote Procedure Call] server on port 10000 [uTorrent Classic] or on port 19575" - and that's not a secret, that's well known - "in uTorrent Web." He says: "There are numerous problems with these RPC servers that can be exploited by any website using" - and then he uses the JavaScript function, it's XMLHTTPRequest(), which is a way for code running on a page to generate requests for various assets. Although, I guess, I think you would need to mess around with DNS because normally you would have some same-origin protections.

But anyway, he says: "To be clear, visiting any website is enough to compromise these applications." And he says in his notes: "As the name suggests, uTorrent Web uses a web interface and is controlled by a browser, as opposed to the desktop application. By default, uTorrent Web is configured to start up with Windows so will always be running and accessible." Essentially it's like a server running in the background. And then, if you want to, you can fire up a web browser and connect to this server running in your machine on the localhost port in order to configure it and drop in requests for it to torrent files.

He says: "For authentication, a random token is generated and stored in a configuration file which must be passed as a URL parameter with all requests." Okay, so that sort of sounds good. At installation it invents a random token, and then that's got to be present in the request URL. He says: "When you click the uTorrent tray icon, a browser window is opened with the authentication token populated." And he gives an example of it, standard localhost URL.

He says: "While not a particularly strong secret, it at least would make remote attacks non-trivial." He says: "Unfortunately, however, the authentication secret is stored inside the web root," meaning you can just fetch the secret by asking for http://127.0.0.1:19575/, which is the root of the server, and it returns it to you. And actually he said: "WTF!?!?," and he repeats that a few times. So he says: "You can just fetch the secret and gain complete control of the service."

Leo: That's pretty funny, actually. So a browser could do that. I mean, JavaScript in a browser, or a website, obviously, yeah.

Steve: Yes, yes, exactly. Just nuts. So the takeaway for our listeners is be very careful.

Leo: Don't run uTorrent I think would be the takeaway.

Steve: Yeah, I mean, I think - yeah. There's no way this is safe because, if you were to go to a malicious website, it could use XML HTTP requests in order to determine the secret token and then issue commands. And he goes a little bit further because he says: "Once you have the secret, you can just change the directory which torrents are saved to, then download any file anywhere" which is writable in the system, meaning you can replace, I mean, you can essentially - a remote attacker who uses this vulnerability in uTorrent can cause your uTorrent instance to go get a file and place it anywhere in the system. So the first one would be something malicious, and the second one would be an autostart invocation in your autorun tree. I mean, it's just crazy what this could do.

So, yes, not safe. And they didn't respond. They didn't actually fix the problem. They just sort of sidestepped it. So this certainly turns up the heat. Certainly anyone using uTorrent, I would say don't leave it running. As Tavis said, by default it just comes up and runs as a service.

Leo: That's how you normally do it, yeah.

Steve: Yeah. So I would say don't leave it running when you're not using it. Use it carefully and be looking for updates from BitTorrent that convince you that this has been fixed in the right way. Or maybe - I think I have a link in the notes to what Tavis wrote. Yeah, I have a link to a 260Blog.com story that talks about this, and I followed that down into Tavis's postings on Google Project Zero. So there is certainly a way to get that. And I would say wait, you know, I would wait until Tavis acknowledges in his blog that they finally got this thing fixed because, yikes.

So in a move that is very worrisome, Georgia, the U.S. State of Georgia, is moving - it's already passed the Senate, it's now in the House - to criminalize independent computer security research. And I don't know if in the last year there have been many podcasts where I have said please, please, please don't let this ever happen. This will be the worst thing that ever happened. So this is a bill to be entitled, is the way this bill reads, "An Act to amend" - this is all legalese - "Part 1 of Article 6 of Chapter 9 of Title 16 of the Official Code of Georgia Annotated." And it says: "...relating to computer crimes, so as to create the new crime of unauthorized computer access; to provide for penalties; to change provisions relating to venue for computer crimes; to provide for forfeiture; to provide for related matters; to repeal conflicting laws; and for other purposes."

And I found the bill. On February 12th, a couple weeks ago, the bill passed the Georgia Senate and is now in the House. The language of the bill says, one, defining unauthorized computer access: "Any person who accesses a computer or computer network with knowledge that such access is without authority shall be guilty of the crime of unauthorized computer access." With no caveats, no exceptions, no research, no responsible disclosure, no academics, nothing, just "if you access a computer or

computer network with knowledge that such access is without authority." Which is sort of the definition of checking the security of something. It's like, knock knock.

Leo: Mind if I hack your server?

Steve: Oh, look, the door just, yeah, the door just swung open by itself. Then, because they're not totally insane, but this doesn't help much, point two in the bill says: "This subsection shall not prohibit a parent or legal guardian of an individual who is under the age of 18 from monitoring computer usage, denying computer usage..."

Leo: Thank god. Whoa, that's good. Oh, that's a relief.

Steve: I know, "or copying data from such individual's computer." And then it says: "All laws and parts of laws in conflict with this Act are repealed."

Leo: Wow.

Steve: There are no exceptions or exemptions for the intent of the unauthorized access, nor for the responsible disclosure of any vulnerabilities which might be found. So as a consequence, honest researchers risk prosecution for verifying. Anyway, as you can imagine, the EFF has blown a gasket over this. They're all over it. There's an EFF branch in Georgia that is...

Leo: This is a law? Or they're considering it?

Steve: This is about to get passed, it appears.

Leo: It's not yet a law, okay.

Steve: No. I saw the voting, it was about two-thirds to one-third in the Senate.

Leo: Yeah, because these people don't know what the hell they're, you know, they don't know what this means. No idea.

Steve: Exactly. Exactly. It's like, oh, god. Yeah. So, again, I mean, I don't know if we're going to - I guess this is where having a federal government comes in, where the federal government creates something that...

Leo: Fortunately, they're so much smarter, this will be no problem at all.

Steve: Yeah, yeah, yeah. Oh, boy. So, I mean, this is horrible, obviously. I'll keep an eye

on it. We'll see if the EFF is able to, you know, maybe it'll be challenged. Who knows?

Leo: Remember, this is the state where it's illegal to teach evolution. I mean, I don't...

Steve: Correct.

Leo: You've got to wonder which industry is pushing this. I wonder if Diebold is in Georgia. Somebody's in Georgia that doesn't want to be revealed.

Steve: Well, and you pronounced it correctly. I have one of...

Leo: Yes, I know, I know, I saw that, too.

Steve: One of our errata is somebody who's really just blown a gasket, speaking of, over me pronouncing "dye-bold."

Leo: I think I said it wrong, too. So I said - that's why I threw it in, just to show you.

Steve: Yup, thank you. So we don't have, as a consequence of the nature of this so-called industry of iPhone cracking, we don't have definitive information from Cellebrite, and we're not going to. But all of the evidence suggests that Cellebrite may now be able to unlock from iPhone 5 through iPhone X, inclusive, all of them.

There was some good coverage in Forbes. Thomas Fox Brewster writing in his security column for Forbes, he says: "In what appears to be a major breakthrough for law enforcement" - and again, with caveats that we don't know absolutely positively for sure, but it's worth just putting it on everyone's radar - "and a possible privacy problem for Apple customers," he writes, "a major U.S. government contractor claims to have found a way to unlock pretty much every iPhone on the market. Cellebrite, the Israel-based vendor that's become the U.S. government's company of choice when it comes to unlocking mobile devices, is this month telling customers its engineers currently have the ability to get around the security of devices running iOS 11. That includes the iPhone X, a model that Forbes has learned was successfully raided for data by the Department of Homeland Security back in November 2017," so last November, "most likely with Cellebrite technology."

He writes: "The Israeli firm, a subsidiary of Japan's Sun Corporation, hasn't made any major public announcement about its new iOS capabilities. But Forbes was told by sources, who asked to remain anonymous as they weren't authorized to talk on the matter, that in the last few months the company has developed undisclosed techniques to get into iOS 11 and is advertising them to law enforcement and private forensics folk across the globe. Indeed," he writes, "the company's literature for its 'Advanced Unlocking and Extraction Services' offering now notes the company can break the security of 'Apple iOS devices and operating systems, including iPhone, iPad, iPad Mini, iPad Pro, and iPod Touch, running iOS 5 to iOS 11.'"

So then Thomas's column goes into much greater detail, citing interesting but necessarily hearsay accounts of various models of iPhones being accessed, but as always without absolute definitive detail, but of course that's because Cellebrite protects these secrets as much from Apple as from their own customers. They could probably put this in some software. They might be able to package it in something that they sell. But Apple would be first in line to purchase it through a cutout and figure out what Cellebrite is doing and, assuming that they really wanted to, fix the problem.

So the way this works is phones must be physically sent to Cellebrite for them to work their magic. And then they either unlock and extract or provide some means subsequently for, after the phone is returned, probably for forensics reasons, to allow the authority that had that device in their possession to then get access to it. So, again, this is hearsay and rumor. Seems to be pretty well sourced, however. So who knows what they're doing, what "in" they may have found. But it does look like, if this is to be believed, there may be a way in, given the sort of extreme access where you have to have - it has to go to Israel in order to be cracked.

Also on the Apple front, following Microsoft with Azure and their Office 365 services, and Amazon, Apple is complying with a new Chinese law which requires the data for Chinese citizens to reside in China. So for all of these companies it's a matter of either complying with Chinese law or losing the Chinese market, which none of them are willing to do. Greater China is Apple's second most important market after the U.S., which generated in its most recent fiscal year $44.76 billion in revenue, which is a fifth of Apple's total revenue for the year. So China's number two, and Apple's not about to say no just because they're not willing to relocate their data.

What has stirred some controversy is that Apple has indicated explicitly that it also intends to store the encryption keys for its Chinese users in China, whereas neither Microsoft nor Amazon have said one way or the other. They've declined to comment. So for privacy advocates, of course, this raises understandable concerns because they worry that China may become heavy-handed once it has the data and the keys under its own control. In some coverage of this, a Beijing-based attorney who was asked about the decision said that Chinese iPhone users are disappointed by Apple's changes to iCloud data storage because privacy protection in China is weak. However, he said that the iPhone users still consider that iPhone is better than some other pure Chinese-made phones for privacy and policy protection, which I think is a sane position to have.

So Apple partnered with a provider, and I had no idea how to pronounce this. It's G-U-I-Z-H-O-U. But the Internet came to my aid. It looks like it's "guay-cho" is the way you pronounce it. And so the company is...

Leo: "Guay-joe."

Steve: "Guay-joe"? You know that?

Leo: Z-h-o is "joe," yeah.

Steve: Oh, very nice. Thank you, Leo. Okay, Guizhou.

Leo: Or "guiy-joe" is probably how they would say it.

Steve: Guizhou, okay, good. Anyway, so the company's name is Guizhou on the Cloud Big Data Industry Company, or just for short Guizhou-Cloud. It's overseen by the government of the Guizhou Province. Apple plans to shift, they said, operational responsibility for all iCloud data for Chinese customers in China tomorrow, that is, by February 28th, tomorrow, although not the data. Customer data will be migrated to servers there over the course of the next two years.

Apple really does sound like it's being as responsible as it can be under the circumstances. They've first of all declined to say when the encryption keys would move, and they began notifying iCloud users in China last month that their data would be moved into China. Updated terms and conditions for China users say that Apple and this Guizhou-Cloud will have access to all data and the right to share, exchange, and disclose all user data, including content to and between each other under applicable law.

Anyway, so the Reporters Without Borders group has urged journalists in China to change their geographic region or close their accounts before tomorrow, worrying that Chinese authorities could gain a backdoor to user data, even if Apple says it won't provide one. And Apple, for its part, again, I think, continuing to do as much as they can, has said that it has advised Chinese customers that they can opt out of iCloud service to avoid having their data stored in China, and that data for Chinese-based users whose settings are configured for another country or for Hong Kong or Macau won't go to Chinese servers. And Apple said it won't transfer anyone's data until they have accepted the new Mainland China Terms of Service. So I think they're doing everything that they can, and this is the nature of a global network and the autonomy that countries have over their own citizens.

Leo: And we should point out that the Chinese could still go to Apple in the current situation and ask for the keys through the American courts. And more importantly, if you're American, this is a clear indication that Apple has the capability of unencrypting your iCloud data and will do so if offered an appropriate subpoena or warrant. Which people, I think, when I mentioned this on TWiT, people went, what? I said, yeah, Apple kind of implies that they're secure and encrypted, but no. Your iCloud stuff, they can see it, just like Dropbox and most cloud storage stuff. And we've talked in the past about why that is.

Steve: Yes. And I think you're very right to disambiguate iCloud from iPhone because what it is that Apple says they absolutely cannot get into is your iPhone. But I remember in the case of even back in the San Bernardino case, if that person's phone had been backed up to iCloud, then Apple said, yeah, we can provide the data.

Leo: Apple literally told the FBI, aw, geez, if you'd just told us ahead of time, we would have told you this.

Steve: Right, right.

Leo: We can't unlock the phone. Now, I wonder, given that Apple's done everything they can to make sure that they can't get into the phone, why they still persist in keeping the keys for iCloud. Probably because, functionally, trust-no-one encryption on a cloud storage service eliminates, as we've talked about before, a lot of features; right? That's probably why.

Steve: Yes, yes. There are lots of things you cannot do that Apple needs to do. If nothing else, probably like recovery. They're now backing up, you're able to back up your iOS devices to the cloud. Then it gets run over by a tractor. Okay, so now you really need it back. So you may need Apple's help.

Leo: Yeah. Customers want it. In fact, we talked before...

Steve: Yes.

Leo: ...about how Apple's kind of stepped back a little bit in the amount of security they have on those phones.

Steve: Yeah.

Leo: Because customers are saying, "Well, I can't get my password. I've lost my data."

Steve: Well, and in fact we will soon be talking about SQRL. And I was teasing people week before last that I would have an announcement to make. It's not something you can download yet, but it is done. I'm in the process of adding a visual monitor screen to the secret data block for us in the GRC newsgroup to watch the app's use of secret data so that we can absolutely visually verify that it is being proactively wiped from memory. The only way I know to do that is just to like actually see it and watch it disappear the moment you're finished with an operation. So during the final wrap-up I need to go through and check the jargon that the app uses, reread the terminology to make sure it's consistent; but, I mean, we're there.

This comes up in this instance, Leo, because the whole point of SQRL is there's no recourse. There's no one, if you refuse to back up your identity, if you refuse to print out something that we help you print out, I mean, if you refuse, refuse, refuse, refuse, then okay. The whole point of it is TNO, you know, that you are responsible. There's ways you can recover your identity, but you really just have to do a couple things in the beginning. Then you're completely recoverable. But this is no third party. And so it remains to be seen whether that is too much responsibility for people to have. Maybe it is. I mean, maybe people don't really care about their security that much. We'll find out.

But what has been built now is a system that at least says, okay, we've built a workable system that is between you and the websites you visit. You can very securely authenticate your identity in a way that breaches at the server can't expose passwords or email addresses or any - SQRL gives websites no secrets to keep so they don't have anything that they have to worry about protecting from disclosure. But the flipside is

there's no third party to go crying to if you forget your master password, although there's lots of recovery available.

So anyway, yes, Leo. So I think you're right. I think that Apple has sliced this carefully and deliberately so that they are able to help people recover the contents of their phones and their photos, like their life's collection of photos, if they need to.

Leo: Right, right.

Steve: And you're always being prompted to increase your iCloud storage. I'm still within the minimum, but just because I'm kind of persnickety about that. I imagine a lot of people just say, oh, sure, I'll pay a little more per month.

Leo: Yeah, Apple pushes you to back up your phone to iCloud.

Steve: Yeah.

Leo: Which means, even if your phone is encrypted, as they pointed out to the FBI in the San Bernardino case, your iCloud's not.

Steve: Yup. Well, it is. But they are able to access it.

Leo: Yeah, it's encrypted. Well, sort of.

Steve: Okay. Now, you're not going to believe this one. Introduced and mandated more than a decade ago, and you travel a lot, so I'm sure you have one, back in 2007, so 11 years ago, all newly issued passports must now be ePassports, as they're called. And I'm sure you've got one, and you've got a little chip in yours; right?

Leo: I do, yeah, little RFID chip in it.

Steve: Yeah.

Leo: It's in the spine.

Steve: Citizens of the 38 countries on the visa waiver list must have an ePassport in order to be admitted to the U.S. And of course U.S. citizens have to have one when traveling abroad. These ePassports have an electronic chip containing cryptographic information and machine-readable text, which makes it very easy to verify a passport's authenticity and integrity. Or at least so goes the theory. The what is it, CPB, the Customs and Border Protection, U.S. Customs and Border Protection, is the division of the U.S. that is in charge of this. And the border staff have long since deployed ePassport readers at most ports of entry because it makes reading the data out of the passports

and automatically and electronically logging the passport data easier, quicker, and more efficient.

Okay. Now get this. Although all ePassport data has been cryptographically digitally signed from the beginning, Customs and Border Protection has never, doesn't today, never has had the software necessary to authenticate the information stored on the ePassport chips. I kid you not.

Leo: So they just take whatever you say, they go, okay.

Steve: Yeah, yeah. So it can be forged at will, and it's super encrypted, super secret, cannot be forged. So they just, I mean, now you're more vulnerable to forgery because this is the unforgeable cryptographically signed ePassport system which doesn't check the signature. Somehow this...

Leo: Well, that makes it easier.

Steve: Yeah, exactly. Oh, those pesky signatures.

Leo: In other words, as [indiscernible] said in the chatroom, fake ePassports are just like fake regular passports.

Steve: Exactly. Well, except worse.

Leo: Feature parity, yeah.

Steve: Because they're impossible to forge; right?

Leo: Yeah.

Steve: So they're more strongly believed...

Leo: Right.

Steve: ...if it's an ePassport because it's all crypto something magical.

Leo: Yeah.

Steve: Except it isn't, and it never has been. So somehow this fell across the radar of senators Ron Wyden - thank goodness he's there - and Claire McCaskill, who wrote to U.S. Customs and Border Protection's Acting Commissioner Kevin K. McAleenan - geez. I

didn't pronounce this head of time. McAleenan. I can't pronounce his name.

Leo: I'm not going to help you. I think it's Guizhou. I think it is.

Steve: It's not even Chinese.

Leo: I think McAleenan is what I would say.

Steve: Oh, thank you. McAleenan. I like [crosstalk]. Yes, Kevin K. McAleenan for clarification of this rumor and demanding some attention. In their letter they said: "CBP does not have the software necessary to authenticate the information stored on the ePassport chips. Specifically, CBP cannot verify the digital signatures stored on the ePassport, which means that CBP is unable to determine if the data stored on the smart chips has been tampered with or forged," their letter stated. And the CBP has been aware of this glaring lapse since at least 2010, when the Government Accountability Office, the GAO, released a report highlighting the gap in technology. So we know what that means. What that means is there was somewhere, back in 2007, there was a public and private key pair that was created. And anyone who is minting one of these passports has super secret private control of this private key.

Leo: Of course they do.

Steve: So the data for the passport is hashed, and the digest, the hash of the data, is then cryptographically signed with the super secret private key, which nobody else has. And every ePassport. And, you know, passports can come from different places, there must be a collection of private keys, any one of which can be used to sign. Okay.

Then, on the border entry side, you would have this device which reads the data, performs the hash of the data, and then uses the matching or one of the matching public keys to decrypt the stored signature and verify that the hashes match. And none of them do that. They never have. So anybody has been able to forge, alter the data in the passports, and it would never be noticed. So maybe some day. It's been 11 years. Maybe we'll get around to doing that.

Leo: It couldn't be that hard.

Steve: Leo, our browsers do it.

Leo: I know.

Steve: I mean, no. It's trivial. I mean, that's, you know, every web browser that all of us have has the ability to verify that certificate that we receive from a website.

Leo: Yeah, can't be that hard.

Steve: No, everybody else can do it except Customs and Border Protection.

Leo: Oh, lord.

Steve: Incredible. I mentioned at the top of the show that Firefox would be losing a feature that some old techie curmudgeons might grumble about. The next release, we're at 57 point something or other right now, the next major release is thus 60. And the Firefox Nightly build, which is currently at 60, has made some changes to cookie management. In other words, it's disappeared. We've been seeing this trend in Firefox for a while. It sort of disappeared from the UI where it used to be present. And then I remember going looking for it, like a year ago. It was like, where did they hide this?

And you have to, like, you have to change the history. It's under your history settings for some reason. And it's like, what? Okay. So you say, okay, like custom history settings. And then out of the UI drops some ability to still see into what cookies you've got. You can browse them. You can delete them. You can basically audit them. That's disappearing. Now, maybe, I'm hoping, that for those who this really upsets, there may be like a browser plugin you would be able to add to recover cookie management if you really have to have it. But I also think this sort of represents a trend in just sort of the general popularizing and user friendly-izing of the 'Net. I mean, what percentage of - I'm sure they have instrumentation, telemetry on this that says, okay, Steve and four other people in the U.S. last month looked at their cookies. So, okay, so not enough to have that still in the UI. So, and we're beginning to see this, where lesser used features are being pruned over time.

So for what it's worth, Firefox moving forward, we lose the ability to audit our cookies. Maybe if there's a, I mean, I'm a little surprised because Firefox has also sort of been the techie user's friendly browser. So it'll be interesting to dig into this deeper and see what the story is. But it has been confirmed that we're losing the ability to manage cookie settings and remove individual cookies in Firefox with the next release.

Under "nothing's certain besides death and taxes," Coinbase has informed 13,000 of its users that their data will be sent to the IRS soon, within 21 days. They fought an IRS order which was first made back in November of 2016, asking for Coinbase's records of all people who bought bitcoin from 2013 to 2015.

Leo: How about selling bitcoin for dollars? That, too?

Steve: Yeah. And what I - I'm not sure.

Leo: Bought bitcoin. You'd think they'd want to know who got dollars as opposed to bitcoin.

Steve: Exactly. So Coinbase informed about 13,000 of its customers who had, now, what the reporting said, completed transactions totaling more than $20,000 through their

accounts in a single year between 2013 and 2015.

Leo: So they're probably more interested in money laundering than tax evasion, it sounds like.

Steve: Yeah. And so they said that they will be complying with an IRS court order compelling it to disclose the details of those previous year transactions. Of course, that's kind of sticky because these are years for which people have already filed their taxes. And so, yikes, you know, you want to make sure that you, well, declared those back then. I don't know what happens if you amend a back return in that way. It's probably going to upset things.

Anyway, in an email and on its website last Friday, Coinbase noted that it had, quote, "fought this summons in court in an effort to protect its customers and the industry as a whole," they wrote, "from unwarranted intrusions from the government." But they ended up losing in court. And so they added that a "...court order requires us to produce information specific to your account." And somewhere I saw, oh, yeah, including taxpayer IDs, names, dates of birth, addresses, and transaction records from that period.

So you're right, Leo. The fact that it's a limit of 20,000 suggests maybe what they're looking for is like major egregious large bulk transfers through Bitcoin and Coinbase. So I guess probably most people would be staying under that limit in terms of transactions. If and when I ever attempt to liquidate mine, I will certainly declare the taxes without question because I'd have to.

Leo: Why take a chance.

Steve: Yeah, exactly.

Leo: Coinbase says it's 13,000 users, so that's a small number, I'm sure.

Steve: Yeah. So Android P is the next version of Android. I guess it's expected at this year's upcoming Google I/O developer conference May 810 in Mountain View. And I was chuckling because it would be nice if the "P" edition stood for Privacy.

Leo: No.

Steve: No.

Leo: No.

Steve: And at least - and do we know, is it always a dessert?

Leo: Yeah.

Steve: Couldn't be Pancake.

Leo: No.

Steve: What's a "P" dessert? Has there already been speculation?

Leo: Oh, yeah. There's lots of them.

Steve: I guess there would be. Okay. Anyway, so of course we've talked extensively about app permissions in Android, about the sometimes overly broad permission requests. In fact, there's a couple people working on Android clients for SQRL, and there was just some recent discussion in our newsgroup about an initial beta of the Android client, or one Android client for SQRL that was asking for more than it looked like it needed just because the framework had those things defaulted. And so he quickly pared it down to almost nothing, which is nice. Anyway, overly broad permissions, about permission auditing and management, and also of course about how background applications, which are loaded, but which you haven't been using for some time, can be used potentially to stealthily monitor and spy.

So according to the Android Open Source Project, AOSP, to a recent commit to that, Google is working on two built-in features that will be part of Android "P," meant to protect its users from malicious apps spying on them using the smartphone's camera or microphone. The XDA developers saw this, and essentially it notices whether - it identifies apps by the UID, the unique ID of the app. And if the app is running in the background for more than a certain amount of time - and that wasn't specified. But probably you switch away from it, and after some length of time, a few seconds, because you're no longer using it, and if you don't switch back within a certain length of time, it loses access to the camera. If the app continues to try to regain access, that will then generate an error, an explicit error from the app.

And as for the microphone, what Google is doing is simply muting the mic. It just returns zeroes in the audio stream so that no audio - the app continues to believe it's receiving, but it's just gone silent, which sort of makes sense because you put it behind other apps, and you're doing something else. So anyway, just nice to see that Google is moving forward and responding to its users' privacy concerns, and in a way that doesn't probably interfere with anybody.

I remember, was it Stuxnet? There was, some time ago, there was some malware that was signed with a valid certificate, and that got us all up in a frenzy at the time.

Leo: Oh, yeah, from Turkey, yeah, yeah.

Steve: Yeah. And remember like some legitimate manufacturer was broken into in the dead of night, and we believe that the certificate was stolen, and that that was then used to sign some high-profile malware in order to allow it to get used. Well, those were the

quaint old days, Leo.

Leo: Oh, no.

Steve: Yeah. It turns out that nowadays malicious code signing certificates are no longer stolen. They are created. There's a threat intelligence firm, Recorded Future, which dug into the underground certificate supply industry, of which there now is one, and came up with some surprising discoveries. First of all, for purchase through several organizations on the DL, you can get valid certificates, validly signed, registered in the names of real corporations, which, I mean, simply by paying some dollars. Most certificates are no longer being stolen from legitimate organizations. They are using the legitimate credentials of unaware corporate entities, and someone is arranging to mimic the corporation in order to cause a legitimate company - examples were Comodo, Thawte, and Symantec - as certificate authorities that were behind signing these certificates that were legitimate, but spoofed.

So essentially what's happening is someone is arranging to spoof the identities of the requests, and non-stolen legitimate certificates are being issued. And because the owning, technically the owning or at least the company of record for the certificate doesn't know that it was done behind their back by an authority in good standing that thought it was doing the right thing, these certificates can live for years, for however long the cert's natural life is, typically two or three years.

It turns out that, of course, as we know, the reason this is being done is that legitimately signed code, it turns out, in various studies that have been done, is up to twice as likely to pass unnoticed through antiviral systems. Antivirus systems, as we know, have become very heuristic in nature. They're making judgment calls, basically. And so one of the signals that they judge on is whether the code is signed. And EV signing, not surprisingly, makes the AV systems even more happy and less prone to give a warning. EV certificates often suppress the SmartScreen warnings in Windows, for example. So signing is extremely effective in obfuscating malware from analysis.

Okay. So what do these things cost? Three years ago, back in 2015, these non-stolen, created-to-order code signing certs first became widely available to the criminal underground. The most affordable version of a code signing certificate can be had for $300. So on the low end is an Authenticode cert for $300. At the high end is an EV certificate with a smart screen reputation rating, which can be had for $1,600.

Leo: That's less than I pay for my EV certificates, my real ones.

Steve: Yes.

Leo: My real ones.

Steve: Yes.

Leo: So why buy a real one when you can get a forged one free? Cheap. Cheap.

Steve: Ah, well, you've got to deal with the dark underbelly of the Internet, of course.

Leo: Oh, okay. But EVs are thousands of dollars. $1699, that's cheap.

Steve: Yeah, yeah. And it comes with a pre-built-in smart screen reputation rating.

Leo: Damn straight.

Steve: Allow it to just get installed by your customers. Unfortunately, it wouldn't bear your company's name. I mean, I would love to know, like, what types of companies. I wouldn't be surprised if they're legitimate-looking organizations so that, if someone did drill into their cert, although no one does - and that's another aspect of our UI which we've talked about before. It's become, you know, they're beginning to be like, oh, no one really cares about who signs their certs. Well, it would be interesting to see what company names are being borne by these fraudulent but legitimately and freshly minted certificates, just to see.

Somehow there would have to be companies where spoofing could be arranged, where all of the verifications which one of, like, a major CA would go about performing could be intercepted or somehow spoofed. Maybe there are insiders inside some corporations that are getting a piece of the action to facilitate certificates being minted in their companies' names. To me that seems a little more likely than going to - because remember, any company can ask for a certificate from one of these issuers. So we have that same sort of "trust everyone" approach to this. If the certificate is signed by a legitimate entity, and the certificate itself is legitimate, then it passes muster.

So anyway, I just thought it was interesting that it's no longer the case that you need to hack or steal an existing certificate in order to get something signed. If you've got the money, and a lot of bad guys don't, they want to do this on the cheap, and so maybe hacking is easier, but you can just purchase certificates now on the dark web.

Okay. Now, in what has got to be one of the cooler hacks to come along, our friend, friend of the podcast, John Graham Cumming. He tweeted, he said: "The beauty of @Cloudflare Workers" - which is a technology that they have, so-called Cloudflare Worker, like threads, essentially - "is that it was easy to integrate @troyhunt's new pwned password service to add a header indicating whether a posted password is pwned or not. Then the server can warn the user."

Okay. So let's back out of that a little bit and understand what this means. So we've got Cloudflare, that is, of course, a terrific CDN frontend, behind which websites operate. What Cloudflare is now doing - oh, I should back up and say that Troy Hunt, who runs the HaveIBeenPwned.com site, he added an API which allows something to query to see whether a specific password is in his, what is it, like five point some million password list. Oh, yeah, 501 million. So not low millions, 501 million unique passwords.

So now what Cloudflare is doing is, when a form is submitted with an HTTP POST, which is the way you submit forms, that contains a username and password. Their edge technology in the so-called Cloudflare Worker looks at the form being submitted, sees whether there is a password in the POST field, and, if so, submits a query to Troy's API to see whether the password being submitted is on Troy's list of pwned passwords. If so, a header, an HTTP request header is added on the fly to that query as it then travels on

to the server running behind Cloudflare's front edge. Meaning that anyone running a website which is hosted on Cloudflare could easily add some technology on their login, or create an account handler to check for the presence of that header, its Cf-Password-Pwned header. If that's there, then that tells any site who's being hosted on Cloudflare that maybe they should consider notifying their user who is in the process of creating an account or presumably logging in maybe if it already exists, that, whoops, this password may not be safe to use. So very, very nifty and cool service.

Troy himself, upon learning about this, he tweeted: "I think this is one of the coolest use cases I've seen for Pwned Passwords yet." He says: "It's a @Cloudflare Worker," he says, "(code that runs in their edge nodes) that can automatically check a password on form posts (such as login) and add a response header indicating pwnage." He says: "That's awesome." So hats off to the guys at Cloudflare. They keep coming up with neat ways to make their service even more useful and terrific.

And Troy, for his part, just launched Pwned Passwords v2 with - and this is where I got the 501 - he says, with half a billion passwords for download. He actually does offer a torrent, and he asks people to please use the torrent. It's hosted by Cloudflare. So if you download this thing, I think I remember it's like 8GB. So if everybody individually downloads the 8GB file, that's a load on Cloudflare's bandwidth. So he says: "Please use the torrent if you can."

Anyway, so last Thursday, in a long and wonderfully detailed posting, and for anyone who's interested, I've got the link in the show notes, but you can just also google the phrase "Troy Hunt: I've just launched Pwned Passwords v2," and that'll take you to his page. For anyone who's interested in this topic, I'm not going to go into it in detail, yet he wrote this fantastic, lengthy sort of update on the whole world of this password pwnage and what he's been doing with it, and the service he offers, and the statistics, all kinds of really interesting and cool statistics that he's collected over time.

I grabbed one, just one paragraph from that. He wrote: "In total, there were 3,033,858,815 occurrences of those 501,636,842 unique passwords." So again, a little over 3 billion occurrences of those 500-plus million unique passwords. He says: "In other words, on average, each password appeared six times across various data breaches. In some cases, the same password appeared many times in the one incident - often thousands of times - because that's how many people chose the same damn password," he wrote.

So obviously - and hopefully this is just historically because we should all now be using random gibberish passwords. People don't. But so what he's got is really interesting statistics. He's done a cross-correlation of password reuse across breaches. And as he said, on average, each password appeared six times across various data breaches, and in some cases thousands of times, not because, well, it couldn't be thousands of users because you've only got one user at most per breach. So lots of password reuse. Anyway, so cool, cool service from Cloudflare and a neat service from HaveIBeenPwned.

And remember that HaveIBeenPwned.com allows you to safely put your password, put your actual password, yours, into the HaveIBeenPwned web page. It is SHA-1 salted and hashed by JavaScript running in your browser, so that password is immediately encrypted, essentially hashed, well enough. And Troy defends his use of SHA-1, which is plenty strong for this application. So it is salted and hashed, sent to him, where it is then securely compared against this half a billion previously disclosed passwords. So you can securely check whether your password has leaked in any of these known breaches in the past. So very, very cool stuff.

As I mentioned at the top, someone tweeting as "Very Stable Genius" tweeted to both, sent to both me and to you, Leo, he said: "Deeee-bold."

Leo: "Deeee-bold."

Steve: "Deeee-bold."

Leo: It's spelled D-I-E. "Deeee-bold."

Steve: Yes.

Leo: Anyway, I was glad to get that corrected.

Steve: And it looks like "Die-bold" to me, but not "Deeee-bold." He says: "As a former employee who has written you several times over the years..."

Leo: Whoops.

Steve: "...with this same request, please, please, I beg you, please" - this is where we're not sure that it was a good thing for Twitter to expand the length of tweets - "stop making me cringe every time the name is mispronounced." Smiley face.

Leo: It's Guizhou.

STEVE "Love the podcasts." Right, Guizhou.

Leo: I'm glad to get the correction because I want to say it right, too. And I actually, it's like GIF and JIF. I say it both ways to cover my bases. So I now know, it's Diebold.

Steve: Yeah, so now we know. And please perk up when you see me stomp on it again. I'm sure that our Very Stable Genius will.

Leo: I'll protect you.

Steve: But anyway, I will do my best. Diebold.

Leo: Steve is a very - Diebold.

Steve: You know, why did they spell it Diebold if they wanted people to pronounce it "Deee-bold"?

Leo: Well, it's probably somebody's name.

Steve: I'm sure it's some guy's name, yeah. Oh, and Michael Horowitz, who writes the Defensive Computing column and has a great site about router configuration, added - this is also in Errata. He said: "Steve, what you said on the last Security Now! about DNS servers was not the whole story." Remember I was answering the question about whether, if you configured your DNS in your router and differently in your computer, who won the war? And I stated that by default "Obtain IP address automatically" and "Obtain DNS servers automatically" was what computers did. So if you changed what was being sent in your router, then that's what everybody would use. But I said you could override that by manually inputting an IP address and/or DNS servers in your computer.

So Michael corrects me. He says: "Peplink routers can override all DNS requests from devices connected to their routers. In this case you WILL use" - you WILL, he has in all caps - "you WILL use the DNS servers in the router, NOT those hard coded in your computer. From the perspective of the router, it's easy," he says. "Just look for outgoing port 53 traffic." And of course he's correct. It's UDP. It's unencrypted. It is instantly easy to find. It's going to be a port 53 query. And so you simply rewrite the destination IP in the packet as it's transiting from the LAN to the WAN outbound, and the router wins in that case. Now, if you got really fancy with something like an OpenDNS, where you were setting up a TCP connection or encrypting your stuff, then that changes the game. But of course then you're using OpenDNS and not just arbitrarily trying to use a different router. So thank you, Michael, for the clarification. And I saw...

Leo: I'm puzzled by this because isn't there a DNS cache in the computer? Wouldn't that be sufficient in the hosts file? Don't those get referred to first? And if they come up with a response, doesn't the computer just stop?

Steve: Well, yes. Okay. So the hosts file is static.

Leo: Yeah.

Steve: And if you do put things in there, it absolutely wins.

Leo: That wins. And a cache wins. If it's in the cache, it's going to win; right?

Steve: Right. But the cache is always started from scratch.

Leo: When you reboot.

Steve: Like when you reboot. And it also...

Leo: But I think it's important because...

Steve: It also times out.

Leo: Ah, okay.

Steve: Yeah, because all DNS will drain over time. DNS records have a TTL, a Time To Live, which is like, eh, sometimes it's a day, sometimes it's a week, sometimes it's an hour. It varies.

Leo: That's the point is, if your cache gets poisoned, it's going to win every time; right?

Steve: Yes, yes, it'll stay poisoned for a while, yes.

Leo: Okay. So, yeah. I mean, I set it in my router, of course. And I think if you set it in the router, barring that, a hosts file or cache poisoning. Well, but then, yeah, the computer setting isn't going to make any difference if you have that kind of router.

Steve: Exactly. And Michael did add another router. I saw another tweet from him. But it passed by, and I didn't catch it. So there are routers, we'll just say...

Leo: That enforce it.

Steve: ...where you are able to override. And I've not seen this in any typical consumer routers, but the Peplink is a router that offers the feature. And the point is from a technology standpoint it's not difficult to do. If the router has the feature, it would be able to intercept anything that was outbound and...

Leo: Ah, and overwrite it, yeah, got it.

Steve: Yes.

Leo: So if had the feature, it'd overwrite anything, including a DNS cache.

Steve: Exactly.

Leo: Well, wait a minute, though. There's no cache query. It's just - no, because the router's not seeing a cache query, it's just seeing 192 dot...

Steve: Right. But the first time the computer...

Leo: The first time.

Steve: Yes, yes.

Leo: So it isn't - yeah. So it isn't that straightforward. The cache wins, and then hosts.

Steve: Actually hosts...

Leo: Hosts, no, I bet cache is first just for speed, Steve.

Steve: You know, in every instance when I've made a change to my hosts file...

Leo: It's immediate? Okay.

Steve: ...it wins instantly.

Leo: Okay. Okay.

Steve: So I think it looks there even before it looks for caching.

Leo: We were talking about Ubiquiti routers, and a couple people asked me about hardware offloading. And so I wanted to talk about that just because it's something we've never discussed on this podcast. And I have a link to a really nice page about it on the Ubiquiti site, for anyone who's interested. And it's significant because it turns out that a long time ago, back in the DDoS era, when people were like doing SYN floods, we talked about how SYN packets, S-Y-N, the synchronized packets that are used to start TCP, they were damaging or difficult to handle because they were so small.

What that meant was that a given amount of bandwidth could carry a very high packet rate. And so it was oftentimes the per-packet handling capability that was the limiting factor. So a router that could handle, say, 100MB of legitimate data could be brought to its knees by much less bandwidth of pure little tiny 60-byte SYN packets because they were so small. That is, there were things that the router did per packet. And if the packets were tiny, many more of them could be sent per unit of time than legitimate regular data-bearing packets.

Well, this is all, of course, well known to router manufacturers. So one of the things that network adapters have long done and that routers do, too, is known as hardware offloading. Because, for example, every single Internet packet has a

checksum. It's a simpleminded checksum. It's a ones' complement checksum, where you just sum all of the packets and don't worry about overflows, I mean sum all of the bytes of the packet. And once you've done that, you stick the checksum in the packet. And the idea is that when any other device receives it, that device sums up all the bytes and verifies the checksum in order to sort of do a simple-minded "were any bits altered in transit."

Well, the point is that takes time. And that's a dumb use of the main CPU, the CPU in our computers. So for a long time checksumming of packets is something that the hardware in our NICs, in our Network Interface Cards or Network Interface Controllers, did for us. It was just like, so the idea was that the driver, the OS driver for the hardware knew that the hardware was capable of doing hardware checksumming. And so it let the packet go right by without bothering itself to fix, as it's building the packet, without bothering to put the checksum in the packet, knowing that the hardware itself will do that with zero overhead. It can do it at register speed so that essentially the software is offloaded from that responsibility.

Well, the same thing happens with routers. The more advanced router hardware has the ability of doing various levels of hardware offloading, in some cases even involving encryption and encryption verification and signature verification of encrypted payloads. And, interestingly, the Ubiquiti routers, many of them, even the EdgeRouter X, has that capability. It may not be turned on by default.

So last week we were talking about a router where the throughput that was measured was - I think it was down in the 400Mb for a given router. And when that listener of ours switched routers to a beefier router, they were able to get right up near the 1Gb that they believed that their connection was able to handle. I did see in this documentation, and again, if you're running a Ubiquiti EdgeRouter that we've talked about, you might want to check out this link. Probably you could google "EdgeRouter hardware offloading explained," and I would bet that Google will take you to the Ubiquiti page, which is very comprehensive. But they have in their firmware the ability to turn on hardware offloading, and everybody will get a substantial benefit if that's done.

Leo: Why is that not on by default?

Steve: Because their page does say that their firmware is a little skittish about it for some reason, and so if there's any compatibility problem created, they want to make sure that users can say, whoops, whatever I did, that just broke something, and so they're able to back out of it.

Leo: I have to try this because I - oh, a couple things on the EdgeRouter X. One, somebody in the chatroom says it does support that Peplink thing that we were talking about of enforcing DNS.

Steve: Interesting. I really do like that little router. It is a kick-butt little router.

Leo: Thank you, Web9462. And I think Neo said pfSense will also do that, of course.

Steve: Yeah, not surprisingly.

Leo: If pfSense didn't I'd be shocked. I actually took my EdgeRouter X out of service because I was noticing periodic very brief dropouts of Internet access. It was just going away. And it was really bothering the Roku. For some reason Roku did not handle it well.

Steve: Oh, well, yeah, I mean, you have a streaming device, it doesn't want to lose...

Leo: Well, there was still enough in the buffer. Well, I might try this while we're offloading. That might actually fix the problem. But taking it out of service and just using a different router fixed it.

Steve: Well, and I was going to say that even if you're not saturating your connections, the hardware offloading could substantially reduce latency. And of course that's the buffer bloat problem, and latency is bad, especially for any kind of real-time streaming stuff. So by all means...

Leo: I don't really need it. So I was just playing with it on your recommendation. So I'm just going to let the...

Steve: So Roku is your choice of streaming...

Leo: No, it's just one of many.

Steve: Oh, that's right. I did hear you say that on one of the other podcasts. You're like, oh, I've got them all.

Leo: I have them all. We often use Roku. On my 4K TV I have the Roku Ultra, Chromecast Ultra, and the new Apple TV. All three do 4K quite well, though the Apple TV is probably the best of the bunch. But Apple TV doesn't support all the channels Roku does, and vice versa. So you need to have at least those two.

Steve: And Leo, Apple TV's controller is the worst thing.

Leo: It's awful, isn't it.

Steve: Oh, my god.

Leo: The Roku is nice, and it has a headphone jack if you want kind of private listening.

Steve: Isn't that freaky? It's bizarre that you plug the headphone jack into the remote control.

Leo: Yeah.

Steve: And you get your audio.

Leo: So I'm very - actually I use the Roku probably more than any of them, yeah.

Steve: Yeah. Apple TV's remote controller is the epitome of the failure of form over function because, oh, my god, is it awful. I just went back to Amazon, the Fire TV, because I finally got tired of fighting with Apple's controller. It's like, it's so bad.

Leo: Yeah.

Steve: I got a nice - actually this is a three-note chain. I don't think I've ever shared one before, but I thought it was kind of fun because it just sort of highlighted a SpinRite customer's interaction with GRC. I've talked about Sue and Greg from time to time. They've both been with me for decades now. A customer named David Stidolph , I hope I - Stidolph? Stidolph? Anyway, David, sorry if I mangled your name, Stidolph.

He wrote on the 17th, he said: "Hello. My name is David Stidolph. I have been a LONG [all caps] time user of SpinRite, but I have an immediate need. I need to recover a laptop drive, and I cannot find my ISO." He says: "Any chance I can download it again?" He says: "My current email is" - and I redacted it from the show notes, but it's a gmail account. So then he said: "My previous email was," and again I took it out. It was a RoadRunner account which he says is no longer active. And he says: "If I cannot download it, please let me know. Thanks, David."

So Sue got his email, responded: "Hi David. Your original receipt with download instructions has just been sent via email. Your transaction code in your emailed receipt is the key to your download. It will allow you to obtain replacement copies as well as edit your own contact email should it need to be changed. Sincerely, GRC Sales Department."

And then he responded in a third mail. He thanked Sue, and he said: "Wanted to pass on a SpinRite story. My first PC was a 4.77" - of course we all know that meant MHz - "clone with a 5MB MFM hard drive." He says: "When it powered up, it sounded like a small jet engine." He said: " I got a copy of SpinRite, don't recall whether it was 1.0 or 2.0, and ran it on that machine. Not only did it recover bad sectors, it quieted the drive down quite a bit." Go figure. He said: "Since then I have saved many hard drives over the years. The software rocks. Best of luck. I hope you guys sell lots. David Stidolph."

Leo: Nice.

Steve: And of course he did receive the ability to download a copy. That's true of all of our customers. We have a database. Sue's able to query the database based on whatever information a customer can provide and can then reenable their ability to download the product. I have a feeling, once 6.1 starts to become available, Sue's going to be very busy helping people to regain access so they're able to download their free upgrade to 6.1.

Leo: Nice.

Steve: I'm not looking forward to that, but our customers are. So a few closing-the-loop bits. Oh, so Stefan Korrivo, whose Twitter handle is @NoThumbBowler. I don't know what that means.

Leo: He has no thumb.

Steve: And he bowls.

Leo: And he bowls. So he's a three-finger bowler.

Steve: A three-finger - okay. Do bowling balls have four holes? I don't know. Anyway...

Leo: I believe, yes, they do. Or three. Three holes.

Steve: So your pinky doesn't go in?

Leo: They have three holes.

Steve: Ah, that's right. That sort of feels right.

Leo: No, four. Three holes. Your thumb and then your index finger and your middle finger.

Steve: Maybe holds the ball with both hands and then rolls it down.

Leo: That's how I do it.

Steve: Yeah.

Leo: Throw it down [crosstalk].

Steve: So he says: "I'm not looking forward to the day that many domains are using different crypto mining and my browser with multiple tabs is rendered useless since each domain will want 90% of my CPU and degrade my Internet experience." Well, the good news is I'm sure when that happens noncurrent tabs will not have the miners running still. So, I mean, this will get managed. I still, I mean, this could be a passing fad, this idea of permitted mining to replace advertising revenue. I don't know. I think it's going to - if it doesn't die before it can be made efficient, I think it might make sense. We'll see. But in any event, the browsers will certainly become aware if this becomes a thing, and it will only be the page your eyeballs are viewing at the moment, which has mining running on it. So when you switch away, that'll be shut down, I'm sure.

Anthony tweeted: "Bluehost wanted me to provide the last four characters of my password for identification. Does that mean they have my plain password?" He says: "I thought password is hashed in the DB."

Well, that's sort of an interesting hack. I meant to add a screenshot that Anthony provided, or I guess I went to get it, or maybe both, because it was interesting. They were in fact asking for the last four characters of his password. And this could be done securely. So when he originally provided them with his password, they would have received the whole thing. They could have hashed the whole password and saved that, and then hashed the last four characters and saved it separately. So they've got two hashes. They've got the full password hash, and they have like a little verification hash stored separately.

Access requires the whole password, but identification - and I don't understand quite the use case of that. I didn't see why it was that they would want identification without full password. But, I mean, it's sort of legitimate. Technically it could weaken the whole system because a bad guy could separately attempt to guess just the last four characters. As we know, there are many fewer possibilities to guess in just four characters. Maybe that would allow them then to spoof this identification phase, whatever that is. Maybe that was, no, it wouldn't be password recovery.

Anyway, I don't know why they would want that. But it could potentially then give an attacker a bit of a leg up because then they could separately verify a piece of the password from the whole and then have those four removed from the whole, although they still wouldn't know what the whole was, then, would then have to guess that. So an interesting hack. But it does not mean, to answer Anthony's question, that the whole thing would have to be stored in the clear. It could have just been set up that way in the beginning, and they saved the last four characters when they were hashing the whole thing.

Leo: I kind of like that idea actually because it kind of splits the baby. You get a way of verifying without them knowing the full password.

Steve: Yup. And you know, for what it's worth, SQRL has that built in, too.

Leo: Oh, really.

Steve: Yeah, we call it the Quick Pass.

Leo: Aren't you smart.

Steve: The idea being that the first time you sit down at your computer, you need to enter your whole password in order to decrypt your identity. But then it immediately reencrypts your identity using just the first four characters. And so that subsequently you're able to just go bing bing bing bing. Oh, and you can change four. You can adjust it up or down to suit your taste. So as you are reauthenticating yourself with SQRL over the course of the next few hours or the day, like whatever, a session, you're able to just give it the first...

Leo: Like a PIN.

Steve: The beginning of your password. A PIN, exactly. And it very quickly says, okay, it's still him. But if you miss, that is, you can't guess wrong once. You don't get those first four right, the first thing SQRL does is wipe that hint data out of RAM and then says, oops, you're going to have to re-prove your identity. But that's just entering your whole password in order to say, oops, sorry, that was just a typo. But sort of a nice tradeoff of convenience and security, I think, one that we haven't seen before, except these guys are doing it for something, which is kind of cool, too.

Stephen Pickering says: "Hey, Steve. I'm in the market for a new Mac, but I hate the idea of buying a new computer that has a patch on its CPU for Meltdown and Spectre, which from my understanding hampers performance. Is mine a valid reticence? Thanks, you're the best." And he also included @leolaporte in the tweet.

So, okay. So it's probably the case that until a next-generation processor with this updated microcode is shipped from Intel, that all existing processors will be patched. And the design pipeline is years long. So it may be two, three, four years before we - or maybe they would be able to update a processor already in manufacturing. I don't know. So you could imagine they could retrofit patched firmware for existing, currently manufactured chips so that they start carrying the patch themselves. But it's not the patch - and this is the key - not the patch that hampers performance. It's the fact that we've had to give up something we had. The patch gives software the control that it has not previously had that is necessary for security. And it's the security change, the security fix which is causing us some performance loss.

So essentially, until recently, and this is why I don't hold Intel responsible, I mean, we've all been - the whole industry has been benefiting from the fact that we were doing something, and "we" meaning everyone, all processors, the entire computing industry, has been doing something that wasn't technically secure and getting a big performance win in trade. Well, now the bad guys have figured out how to leverage that insecurity, that vulnerability, and so we're having to back off of some performance that was nice to have, but now unfortunately we can't have it anymore.

So future processors will incorporate the patch, and software will use that in order to throttle performance a little bit in the name of security. But we were sort of getting away with something for a long time that was fine until the academic researchers said, whoopsie. Actually, it was Google's Project Zero said, uh, Intel, you know, we can read RAM in other processes. Did you know that?

Leo: Apple says that benchmarks show almost - or let's see the exact - no measurable performance reduction for their Meltdown mitigations. They did the Spectre mitigations in Safari. And on two Safari browser benchmarks, one has no measurable impact; one has less than 2.5%. That's the JetStream benchmark.

Steve: Good.

Leo: So I don't think you're missing a whole lot on a modern Mac. And that's the point. In fact, they aren't even fixing the older Macs. I think they're just going to let those go because that's where you'd take a big hit. By the way, it's not just Macs, it's any PC. I mean, I think you'd have some reticence on any PC, any modern PC; right?

Steve: Yes, yes, yes.

Leo: But I think Apple's done a good job, and it sounds like at least they don't believe it's much of a hit.

Steve: Right.

Leo: Modern PCs are so fast. You know, you wouldn't notice. You wouldn't know the difference.

Steve: Oh, I know, Leo. But we're so spoiled.

Leo: Make sure you get an SSD. That makes a bigger difference than anything else.

Steve: Yeah, exactly.

Leo: And lots of RAM.

Steve: Yeah. So a few mining questions, which leads us into our discussion of WebAssembly. Nick Stoler said: "Regarding JavaScript mining, wouldn't WebGL allow pretty efficient mining without building it into the browser?" And I'll respond to that because that's, well, in fact then Michael, who tweets from @Krabby127, said: "On Security Now! you keep mentioning how browser-based mining are currently terrible because JavaScript is inefficient. What about WebGL? That leverages the GPU directly. Love the show."

Okay. So both Nick and Michael, WebGL does give browsers access to the graphics processing unit, but not for raw hashing, which is what we want. The WebGL accelerates, well, WebGL primitives, making them run much faster. So graphics-y things like blitting areas or fast vector computations or 3D permutation and matrix multiplications, those

primitives which high-end graphics processing needs, that's what's accelerated on the GPU. What mining needs is just raw brute-force hashing. And so that will be the innovation that I hope we'll see is that there will be some extension of web hash or web crypto or who knows what we'll call it which gives browser code access to GPUs for hashing. That will be the thing that then gives us really good performance and leverages the GPUs that are sitting around looking for something to do most of the time, relatively speaking.

So finally, two of our listeners, Sebastian, I hope this is Boisvert? B-O-I-S-V-E-R-T. Probably maybe the "T" is silent? I don't know. Anyway, Sebastien, sorry I mangled your last name. So Sebastien said: "Would WebAssembly make web crypto mining more practical?" Andreas asked, he said: "Second week in a row you talk about JavaScript being way too inefficient for crypto mining. What about, for example, WebAssembly?" And then he gives me the URL, WebAssembly.org.

So I wanted to address this, just to wrap up this week's podcast. WebAssembly is better than JavaScript, but still worse than native code. It is a standard. It is definitely on the way to happening. The first benchmark is called the MVP, the Minimum Viable...

Leo: Usually product, sometimes program.

Steve: Minimum Viable...

Leo: Product. Or Program.

Steve: Protocol? I can't remember what "P" stood for in this case, MVP of WebAssembly.

Leo: Yeah, Minimum Viable, well, hmm, okay.

Steve: Protocol?

Leo: It's very commonly used in startups, just Minimum Viable Product.

Steve: Right. So what they're doing is they're - okay. So WebAssembly - and Leo, you'll connect to this because I know you've always been interested in stack machines like Forth is. WebAssembly is a stack-based virtual machine. So it is not the - there was the experiment that we talked about a few years ago that Google did where they were using a restricted subset of Intel assembly language, or Intel machine language actually, which they had managed through an amazing amount of work, I mean, just like I was so in awe, where they were sort of able to sandbox native Intel code and arrange to keep it from doing anything dangerous.

Well, that was cool because it was then running on the bare metal. I mean, it was actually native code. Well, the problem is it was also Intel chip specific. And in this day and age, while Intel is the current leader, as we know, ARM is on the rise. ARM isn't going away. And so it didn't make sense to standardize on a single manufacturer's native code. So what WebAssembly is, is it is a binary format, meaning that it is binary code

that is downloaded. It also has an assembly code-looking text format, and it is executable in web pages. It's becoming universally supported, is now in the hands of the World Wide Web Consortium with engineers from Mozilla, Microsoft, Google, and Apple. So it's going to happen.

It is compiled currently from C and C++. So you start with C and C++ code. However, it can be targeted at other high-level languages in the future. So C and C++ compiles to WebAssembly, but it is bytecode, as the term is. It is interpreted bytecode. The reason it's interesting is that it is very compact. It runs fast. And it's, I mean, the goal is to approach native machine code speed. It isn't there yet because, again, you need an interpreter. So there's a layer of interpretation which is emulating a virtual stack-based machine which reads the bytecode and does things with it. So there's still a layer of translation there.

But for web browsers, the reason it really wins is that it is very small, so it downloads fast. And it can start running the instant it arrives. JavaScript is ASCII. It's script. It's text. So we've got all of these JITs, all these just-in-time compilers and JavaScript interpreters. And to their credit, they're doing an amazing job. But the code comes down as text and then has to essentially be interpreted/compiled. And, I mean, what they've done is an amazing amount of work. But think about it. Every single time some big Node.js blob is downloaded, or any JavaScript is downloaded anywhere, it is going into a browser, and all of that work is having to be done, redundantly, by every single person that downloads it.

What WebAssembly has done is, I think very cleverly, they've moved that work to the other side of the connection. They've moved it to the delivery side, or to the server, rather than weighing down the recipient. So code gets written in a comfortable high-level language like C. It's compiled into WebAssembly, and it's a binary - it's a precompiled, ready-to-run, binary blob. Yes, it's not native code. But the advantage of that is it can run on any processor that has a browser that runs WebAssembly, and they're all going to.

So what we're going to see is I think we will over time see a shift away from JavaScript to WebAssembly because it'll just make sense. What the user experience will be is snappier applications. It'll just, the moment the page is there, the code is running. It's not running at native speed, but it's running faster than JavaScript. But mostly it's starting up instantly. So it makes a lot of sense. So it doesn't - it gets us closer to the hardware. Still, in order to have chip independence, it's going to be a virtual stack machine, which is now well defined. It may be moving forward in the future as it evolves, so there will be Version 1, Version 2, and so forth. But what we really need is we need GPU hashing support in our browsers. When we get that, either by an extension maybe or native in the browser, that'll give us high-speed crypto mining capability, managed then by the browser.

WebAssembly, nice step forward. It is happening in the industry, and it's probably a better solution for mining than JavaScript, but not the same as running native code. So that's the tune-up on WebAssembly. We haven't really talked about it much. We talked about it once before, but it's really - it's moving forward. And it's going to end up becoming a substantial, I mean, I think it's going to largely replace JavaScript in the future because it just makes sense.

Leo: And it's server-side, not client-side.

Steve: Correct. And so, yeah. So like right now with JavaScript, everybody who downloads a page with JavaScript has to essentially compile it in order to run it. It makes much more sense to do that once, like for the web developers to do that once, and then just offer a much smaller precompiled binary which downloads and then just drops in and it's ready to run on the WebAssembly virtual machine, essentially. Very cool.

Leo: Hmm. Interesting idea. And you can find out more about it. They have a WebAssembly.org page.

Steve: Yup.

Leo: Where they describe the MVP, Minimal Viable Product is what they've got.

Steve: Ah, okay, perfect, thank you.

Leo: It was the first conceptual version of it. And that was the Tank game, if you were watching on video, I was playing. It was written in WebAssembly, and it was pretty snappy, using WebGL.

Steve: Oh, very cool. Neat.

Leo: Yeah, yeah. Well, Steve, we've come to the end of this fabulous, gripping edition of Security Now!. I've been gripped.

Steve: Gripped? Hold onto your whatever you need to grip.

Leo: I hope you've been gripped, as well. We do this show every Tuesday, right after MacBreak Weekly, around about 1:30 p.m. Pacific, 4:30 Eastern, 21:30 UTC. I'd love it if you stop by and say hi. You could either do that at TWiT.tv/live, that's the live video stream. You can listen on any of your voice-enabled devices, just ask for TWiT Live. The syntax depends, and it's changed, apparently. On Amazon Echo you have to now say "Listen to tune in TWiT Live" for some reason. But that works. You can also be in-studio. Email tickets@twit.tv, we'll make sure there's a chair out for you.

And if you can't watch live or be here live, you can always see the on-demand product. Steve keeps it at his website, GRC.com. That's where you can get the 64Kb audio plus the nice transcriptions that Elaine Farris does, so you can read along as you listen, or search. It's a great way to jump right into the show. We have audio and video at our site, TWiT.tv/sn, for Security Now!. And Steve's @SGgrc on Twitter. I know a lot of people talk to him that way, so that's a good way to reach him, or GRC.com/feedback.

While you're at GRC, please pick up a copy of SpinRite. You'll get a copy of 6.1 the minute it's done. You can also read about SQRL, all sorts of stuff, GRC.com.

Everything's free except SpinRite, but that's the one you want to pay for, so there you go. And Steve will be back next week to discuss the latest in security. Thank you, Steve.

Steve: Yes, sir. Who knows what's in store for us.

Leo: What evil lurks in the hearts of hackers. The Gibson knows.

Steve: Catch up next week. Thanks, buddy. Bye.