SECURITY NOW!

Transcript of Episode #742

Pushing DoH

Description: This week we look at some interesting changes coming to Android and some inherent challenges presented by the nature of the Android ecosystem. We examine some newly revealed troubles with the venerable VNC clients and servers. We note a welcome change to Twitter and update on law enforcement's "foregone conclusion" strategy to force password divulgence. We then look at a surprising pre-announcement from Microsoft about DNS, then dig more deeply into the details of the emerging DoH protocol and reveal a VERY interesting and surprising and unsuspected capability.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-742.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-742-lq.mp3

Leo Laporte: This is Security Now! with Steve Gibson, Episode 742, recorded Tuesday, November 26th, 2019: Pushing DoH.

It's time for Security Now!, the show where we protect you and your loved ones, virtually, online, your privacy, too, with this guy right here, Steve Gibson of the Gibson Research Corporation. Hello, Steve.

Steve Gibson: Leo, great to be with you again. I'm actually going to be with you again.

Leo: Yes.

Steve: In a few more days.

Leo: I'm so excited.

Steve: Weather allowing.

Leo: It's pouring rain here right now.

Steve: Yeah. Lorrie was thinking maybe we should drive, and I said, oh, you know, I don't think a little rain is going to ground the airplanes, so...

Leo: No, they know how to fly in the rain.

Steve: Yeah. And I figured that all of these cancellations we're hearing about are in the Northwest and also the Northeast, where it's snow and ice and sleet.

Leo: Yeah, it's not rain, yeah.

Steve: Santa is thinking, maybe we can move Christmas to - how about March?

Leo: Yeah, maybe, maybe. Yeah, Saturday we're doing a special. I should mention, so everybody knows, we're going to finally take the wraps off SQRL.

Steve: Well, in the official TWiT studio wraps, yes. I've had a few rehearsals, and I'm ready for...

Leo: All over the world.

Steve: ...the grand finale.

Leo: Steve's done his world tour, and so it'll be me and Steve. Is there anybody else you want? There was some - I think somebody said Father Robert might be in town, if you want to have him on, too. I do know we have 50 people in the studio. We have a very full house.

Steve: Well, that's good. And really what my deal is, is just sort of a one-man walkthrough of the - I call it "The SQRL Story," like what happened, what was the original spark, and then how it evolved. And then why it took so long was all of the other details had to get resolved, like what happens if this, what happens if that, what happens if. Anyway, so I basically explain, lay the entire thing out, explain the entire thing. So we can certainly do, and it would be good to have some time for some Q&A at the end. But it's really sort of a presentation, rather than interactive. Otherwise we just get tied up in details and off the track.

Leo: I agree. I would just derail you. So, good. Well, there'll be plenty of people to ask questions. So that's not going to be problem at all. And then we're going to be getting together at Lagunitas afterwards. We're going to have a little meet-up.

Steve: Yes.

Leo: So that'll be fun, too.

Steve: Yes. And it's fun to have just the crowd dynamic, instead of like talking into a dead camera.

Leo: Oh, yeah. Now, I should tell people, don't just show up. Everybody who's coming has already RSVP'd and asked for tickets. I should also mention there's a long wait list. However, if you're not coming, but you got tickets, please email tickets@twit.tv and tell us because there are people on the wait list who would love to be in the studio audience.

Steve: Once upon a time I belonged to a video dating service. And they had on the front of the VCR, it said, "Be kind, rewind." And that sort of relates here. Don't leave your seat empty. If your seat's going to be empty, just let TWiT know so that we can fill it with somebody else's butt.

Leo: There's plenty of people want to come to this event. It's very, very exciting. All right. Now, what have you got for this episode?

Steve: This is a tease title because the title is "Pushing DoH," where of course "DoH" is DoH, which we know is DNS-over-HTTPS. But the way I'm using the word is unexpected, so our techie listeners can say, hmm, what does he mean by that?

Leo: What does he mean by that? Hmm. Hmm.

Steve: Pushing DoH. We're going to find out.

Leo: All right.

Steve: Because I was led into this further by a surprising preannouncement from a major OS vendor, which I thought, oh, that's odd. Why are they preannouncing? I have a sense that there's politics going on behind the scenes. But that caused me to do a little more digging, and I discovered several things that we've never talked about before that sort of changed the territory. And I'm guessing that the future will look different than we may think. So we're going to wrap up with that by solving that mystery.

But we have some interesting changes coming to Android that we're going to talk about, some inherent challenges presented by the nature of the Android ecosystem which have come to light as a consequence of a recent report. We've got some newly revealed troubles with the venerable VNC clients and servers. I'm sure back in the day, Leo, I know I did, you probably were using VNC.

Leo: Oh, not back in the day. Still. Oh, gosh, yeah. We even recommend people continue to use it.

Steve: Yes. And in fact it is heavily used, especially in the industrial enterprise.

Leo: Oh, VNC, not VLC. You're talking about remote access VNC.

Steve: Yes, yes, yes, yes, VNC.

Leo: Yeah, my favorite VNC client was a Mac client called Chicken of the VNC.

Steve: Okay.

Leo: So you're right, I haven't used that one in a while, yeah.

Steve: Well, over in Windows we got UltraVNC, TightVNC, and a whole bunch of other things. Anyway, turns out a group took a look at VNC clients and servers; and, not surprisingly, it's time to update.

Leo: Uh-oh. Uh-oh.

Steve: If you haven't. We also have a welcome change to authenticating with Twitter that I just wanted to make sure, because I know we have, I mean, I know from my own Twitter feed how many of our listeners are Twitter users. And there's good news on the authentication front, and it only took Jack Dorsey being hacked to make this happen. We've also got an update on law enforcement's foregone conclusion strategy. That was something that Jason and I talked about that you missed while you were...

Leo: Oh, I'm well aware of it. It's doctrine, not a strategy, that allows them to get your password.

Steve: It does. And so anyway, so there's been some update on that. We also look at this, as I mentioned, a surprising preannouncement from a major OS vendor about DNS. And then we're going to dig - that sort of led me to dig more deeply into some details of this emerging DoH protocol that we've been talking about because of its wide application in browsers. I think our listeners are going to find it very interesting.

Leo: Can't wait.

Steve: So I think another great podcast.

Leo: I am very excited.

Steve: So our Picture of the Week I've had for a while. One of our listeners processed some number of transcripts and did a word frequency analysis and then created a word cloud graphic. And, I mean, and nothing really stands out. Apparently "know," K-N-O-W, and "just," J-U-S-T, are right up there along with "like" and "that's." So those are rather generic.

Leo: But that makes sense. The bigger the word in the word cloud, if you're watching the video, means it's used more frequently.

Steve: Yeah, yeah. But we do have "Microsoft" down there at the very bottom. We've got "machine." There's "Leo," your name over there on the right. "SpinRite."

Leo: I see "SpinRite." I see "password."

Steve: Yeah, and "Internet," oh, and "security," and "people."

Leo: I know this is old. Do you know how I know this is old? There's no SQRLs.

Steve: Yeah, that's a very good point, yeah.

Leo: We've got to redo this.

Steve: Yeah. So I have had it for a while. But anyway, I just thought it was kind of cool. I had been sitting on it and hadn't shared it, so I figured, well, what the hell. It's be fun just to look at it.

So there are some recent rumblings about Google's possible return to the stock Linux kernel for future Android-based smartphones. And then there's also some different rumblings about an entirely different new OS that's being developed inside Google. We've touched on it. I remember mentioning it a couple years ago. It's sort of been on a low simmer. People are still working on it, apparently a hundred Googlers, and that's Project Fuchsia. But it appears to be still some number of years away.

John Dunn, who was writing for Sophos, recently noted that the momentum appears to be building inside Google for what looks like it would be a radical overhaul of what he described as Android's tortured relationship with its precious Linux-based kernel. He wrote: "It's a big job and has been a long time coming, arguably since the mobile operating system was unveiled in 2007." And I had to do a double-take. It's like, 2007. It's been 12 years that this has been going on. But, yeah, we've had Android with us for quite a while.

So the company hasn't made any firm announcements. But journalists recently noticed a low-key video posted to YouTube of a presentation given by the Android kernel team chief. His name is Sandeep Patil. And that occurred during September's Linux Plumbers Conference in Lisbon, Portugal.

So what's wrong with the way things are today? We've talked around this a lot because Android security, keeping Android up to date is a real problem. The development model that underpins how Android uses the Linux kernel, that has sort of evolved, inherently leads to a lot of complexity, which slows down updates, raises costs, and makes life difficult for both Google and for the device makers downstream in all sorts of ways. The result is that the Linux kernel used by an Android device can be slightly different for every make and model and at different points in time.

The device makers start with the LTS, the Long Term Support kernel, before the device's so-called "out of tree" Android common kernel customizations are added to that original based Linux kernel. And according to Patil, there are many of these. They've been working to reduce them just because at one point it was really getting out of hand. But post-reduction, as of February of 2018, 355 changes needed to be applied which required 32,266 code insertions and 1,546 code deletions on top of the base LTS v4.14.0. And as I said, even that much tweaking was an improvement over the way things had been. After that is done, then the system on chip companies like Qualcomm add a bunch of their own hardware-specific customizations. Then manufacturers add even more of their own vendor and device-specific software.

So the consequence of this is understandably something which is very difficult to move. And once you've got this built, you're really disinclined to start over, which of course is necessary when various types of problems, specifically vulnerabilities in the kernel, are found. So what we end up with is multiple sequentially applied layers of customization which require that each device use a root kernel as its starting point, but then all of those changes flow downstream and affect everything else.

So of course we end up with Android devices that tend to have a limited shelf life. But most problematically, it makes the application of anything we want to change, any patches, much more time consuming, labor intensive, error prone. And, I mean, the human factor consequence is that it just doesn't happen as smoothly and easily as we know security updates have to be applied.

So what Google appears to be planning is essentially to scrap this whole approach, to return to the use of a base generic Linux kernel, with the goal of eliminating, if possible, the need for individual Android kernel modifications. So what we would in effect have is an Android Linux kernel, which doesn't get changed, but then on top of which well-defined modules would be placed. Essentially, sort of fixing the architecture. You end up with the classic layered architecture with a well-defined interface among layers, rather than what has sort of evolved over the last 12 years, which is, oh, let's just go and apply a patch. Oh, and let's apply one over here. And, oh, now we need one over here.

Well, we ended up with 355 of those, which required all kinds of modifications. And you can imagine when the source changes you have to reinspect all of the things that you're wanting to change and update them in order to correspond to what the source has done. So anyway, it's looking like Google is in the process of completely changing the way this operates. And this is good for us because, Leo, I'm sure you would know, the total Android OS base...

Leo: Oh, it's got to be billions.

Steve: It's the largest OS in the world; right?

Leo: Yeah, it's got to be, yeah.

Steve: I mean, because of all of the numbers of devices which are now Android-based.

Leo: This won't affect 90% of them, though, because almost all of those devices don't get updated. It's only the new devices that get updated. So even if they change to a different kernel, only new devices will get that.

Steve: Well, the idea being, though...

Leo: Oh, but from now on this will improve things.

Steve: Yes. Yes.

Leo: Absolutely. Google's been working on that with Project Treble and other things. They've really got to do that, yeah.

Steve: Well, yeah. And in fact in my notes here I mention Project Treble, the Project Treble API which we talked about last year, whose goal was to speed up device patching. And so this is in line with that.

Leo: Basically, what Treble does is it puts patches in the store so that we no longer have gatekeepers. The problem has been manufacturers and carriers are gatekeepers, and they block them. But if you put it in the store, then the individual updates it, and it gets updated, yeah.

Steve: Right, right. And we don't know really where Fuchsia is. I mean, the ambition appears to be big. It looks like, if the rumors are true, it could replace Android for the smartphone platform. It is expected to replace the Chrome OS, and they're looking at the desktop, as well, which is, you know, you and I have talked about the idea, for example, of China rolling out an OS from scratch on their own. I just, I think, wow, that's - why would you do that in this day and age?

Leo: On the other hand, I mean, when's the last time we had a new OS? Android, 2007. And it wasn't even really a new OS because it's Linux. I think there's something to be said. Look at all the problems we have now with Windows and iOS and macOS with bugs. And I think part of it's the fact that these OSes have been around so long, they're just patches upon patches upon patches. I think the idea of starting with a clean slate is not a bad idea. We've learned a lot in the last 20 years.

Steve: Yes, and I would amplify that and say it's always a good idea. I mean, the OS of today has a very different architecture. I mean, for a while there was the whole jargon of the microkernel. The idea was that you just had this tight little perfect microkernel. And then instead of having a massive kernel, you would keep everything else out. Well, that was even the original NT design. But Microsoft could not help themselves. They said, oh, but look at all those ring transitions that the GDI is making. Oh, oh, oh, oh, please, please, please can't we just put GDI in the kernel? It's like no, no. But they did.

And as a consequence, now you're able to have a JPG take over your OS as a consequence of the fact that the rendering is being done down in ring 0 rather than the original, you know, the ivory tower concept was, oh, no, those will all be unprivileged processes that operate as services which the kernel calls. And that way if anything goes evil in them, it can't do any damage. Well, nice idea. But it didn't survive because we're all too performance happy. We just had to have the performance.

Well, now here we are again, and we sort of talked about this relative to WebAssem last week, where the WebAssem guys realize they've got so much performance with a what-has-not-yet-been-adulterated model that put security first that, oh, if we can please just not debauch it, then we could end up with a system which is secure by design, which we don't see around us anywhere.

Leo: Right.

Steve: Because we always have to just cheat and say, oh, well, but gee, look at that performance over there that we're missing. And people say, uh-huh. But we did that on purpose; remember? And it's like, oh, no, we fired him.

Leo: Oh, no, unh-unh.

Steve: He kept storming into all of our meetings and screaming about how we can't...

Leo: He was driving us nuts.

Steve: Yeah, well, we got rid of him, so now we're going to have more performance.

Leo: Well, no, I think this is interesting and good, although as you point out it's a nontrivial thing to do, so...

Steve: It is really. It's like, oh, boy. So Kryptowire released a report where - and it was a self-serving report. I mean, they have an app which scans firmware for flaws. On the other hand, they scanned a whole bunch of Android devices. And what they found does teach us a lesson. So as we know, Android is the number one smartphone operating system. And whereas Apple has the powerful advantage of having total control over relatively few different hardware platforms, and where related platforms differ, it's often in little details like the screen resolution. They're not needing to deal, Apple isn't, with an essentially infinite number of hardware variations. And there's only the one vendor, Apple. That is to say, they are absolutely vertically integrated. They're doing all of the hardware design. It's operating exactly the way they want it to. They're able to intimately integrate their software to it. And it's all under their control.

Thank goodness for Google because we don't want to have a mono culture. We want, I mean, look at what Android has brought to us by essentially creating a platform that allows far more diversity and also far more affordable solutions. I mean, I look at people who are toting Android phones around. And I know, Leo, you like them because of the freedom that they offer you, and great cameras, and lots of features. But there's also, you know, so there's like the high end. There's Samsung and Huawei and Xiaomi. But then there's also obscure suppliers. And I didn't even - they were in this report, so I thought, who's Bluboo and Leagoo and Ulefone and Walton? Those are brands I'm not aware of, but they exist.

So what's been created, of course, is this very rich ecosystem. But securing it is a problem. Kryptowire, well, they start off their report, their first couple lines reads: "Pre-installed apps and firmware" - that is, okay, pre-installed apps and firmware, that's really the key takeaway - "pose a risk due to vulnerabilities that can be pre-positioned on a device, rendering the device vulnerable on purchase. To quantify the exposure of Android end-users to vulnerabilities residing within pre-installed apps and firmware, we analyzed a wide range of Android vendors and carriers using devices spanning from low-end to flagship. Our primary focus was exposing [what they described as] pre-positioned threats on Android devices sold by United States (U.S.) carriers, although," they said, "our results affect devices worldwide."

I have then a page here in the show notes with two charts. First of all, they found in the selection of devices 146 CVEs in brand new, like just take the wrapper off, phones across the spectrum. 41% of the problems they described as system properties modification. Oh, I'm sorry, no, 41 of the problems, 41 of the 146, which represented 28.1%. They found 26 CVEs, which was 17.8%, in wireless settings modification. 30 problems, 20.5%, which allowed command execution on the devices. There was a little itty-bitty sliver that was AT command execution. I remember, Leo, when you and I talked about that quite a while ago, we were surprised that AT commands were still in there, you know, Hayes will never die.

Leo: It never goes away, yeah.

Steve: Never goes away. There were also 34 problems, 23.3%, app installation flaws. Eight problems were found to allow or affect audio recording. And 4.1%, and there's no number there, so fewer than eight, probably looks like six or seven, dynamic code loading. So sort of flaws across the spectrum. The second chart is a histogram, a bar chart showing which classes of flaw. There were two: exploitable by a local app or exploitable by system or signature app. And what was surprising was the company that had, out of the shrink wrap, highest number of CVEs found was Samsung with, like, 30, looks like maybe like 33 out of the 147 total. Second was ASUS. Third was Xiaomi, and Tecno, no, Lava looks like fourth, and Tecno was fifth. Anyway, so, and I know, Leo, because I've heard you complain about one of the annoyances with the Android devices is all of the stuff, all of the so-called "value add" that is added garbage...

Leo: Right.

Steve: That is added to these things in order to say, oh, look at all these extra bells and whistles we have. Well...

Leo: I wish they broke this up a little better because, as you mentioned, red is a local app. Blue is either the system or a signature app. And I'm guessing that's those Samsung apps.

Steve: Correct, yes.

Leo: And so we don't know if it's a Samsung app or a problem with the operating system.

Steve: So, yeah, right. What we do know from the report is that it's not problems with the base of the device.

Leo: With the Google stuff, right.

Steve: Exactly. And the fact that there are devices with so few problems, well, those are devices that are more bare, that they didn't bother to add any junk to them, any high-end customization, you know, gee-whiz features. They just shipped a base Android device.

Leo: This pie chart, though, not to throw you back in time, but I can't really figure out - okay. So 41% system properties modification. But then underneath it says 28, oh, no, 28.1%, sorry, 41 of them. What does that mean? Is that how the hack was? I don't, yeah, I don't get what they're talking about.

Steve: Well, no, it's what it was that the hack was able to do.

Leo: To do. Oh, so this is the effect of the hack.

Steve: Yes. So the vulnerability - exactly. So it was able to modify properties of the system, which it shouldn't have been able to get to.

Leo: Yeah, okay.

Steve: So anyway, they explained. They said: "Devices are shipped with pre-installed software. The apps are not present on or vetted by official app stores. Most of the functionality is built-in and cannot be disabled. And the apps run with privilege and system access by default."

So what's happening here is that Google, at the base, is going through all of this effort to create a secure platform. Then the vendor that wants to distinguish themselves and their device adds a bunch of custom stuff, and the custom stuff has bugs. And the apps run with privilege and system access by default. So essentially we've got a hardware vendor who is adding software that some guys wrote for them, you know, some of their software people said, oh, yeah, we'll write some apps. But they're not professional security people. They're "whip out the app under deadline and add it to the phone."

And it turns out that those are introducing vulnerabilities into the device that then is able to get in and undermine the security of the rest of the Android phone, as a consequence of the fact that a vendor in this rich Android ecosystem just decided to add some of their own stuff. There is the potential for remote and local exploitation. They found instances where backdoor functionality and data exfiltration was possible, all because essentially bells and whistles are added in order to make the device more attractive on the market. But they're not being done with the kind of eye that we know security needs because it's hard to get that stuff right.

Leo: Wow. That's a real argument, then, for buying a Google phone over a third-party phone.

Steve: Yes. And so I'm not familiar. Do the Google phones, are they only the base phone?

Leo: Yeah. They're stock.

Steve: And then you add things from the Google Play Store.

Leo: Yeah. But remember, of course, every Android phone ships with a lot of Google software on it. But that's the software you're getting on a Google phone, rather than Samsung's or LG's or anybody else's, or ASUS, or ASUS's [crosstalk].

Steve: Right, right. Well, and, I mean, we know how difficult it is to do professional secure software. If someone just whips out some WYSIWYG fancy launcher for an off-brand phone, it's more likely than not going to have problems.

Leo: Thing is, everybody installs - I have hundreds of apps, third-party apps installed. I guess the difference is they don't have system access in the way that a Samsung app does.

Steve: Exactly, exactly.

Leo: Plus Google scans all the third-party apps that you get in the app store and so forth.

Steve: Exactly. And so these things are built in. You can't remove them. And they never went through any app store scan.

Leo: That's what I really hate, yeah. I just really hate that.

Steve: Yeah.

Leo: So that's - I've always been prejudiced against third-party Android. I really like Google Android.

Steve: Yeah. So VNC users. Time to update. Kaspersky's researchers found 37 vulnerabilities, looking at four VNC implementations: LibVNC; UltraVNC; TightVNC, and they noted 1.x; and TurboVNC. Kaspersky did responsible disclosure, notified the developers, and most but not all of those 37 identified problems were fixed. As our listeners know, being a Microsoft user myself, I use Remote Desktop, although only with it safely ensconced behind several layers of traffic filtering firewalls. You will not find a listening remote desktop port anywhere within GRC's IP space. That just - it doesn't exist. But in years past I've played with UltraVNC and TightVNC.

I remember that - you remember Bob Basaraba, my friend, my Canadian friend. He was just - he had TightVNC installed on every computer of all of the people - I would call them his victims; he would call them his clients - who he was managing. And I remember that one of the things about VNC that was cool is that with Remote Desktop the client is always connecting to the server that shares its desktop or that provides a virtual desktop environment. With the VNC tools, you can invert that relationship so that it's possible for the user who is the client to call to the server that is listening, and then they are sharing their computer. So it's more like the remote service model, which of course is the reason that Bob was doing that.

Anyway, they're still around and going strong. And as I mentioned, they have a very strong following in the industrial controls environment, where you are typically VNCing into some sort of a user interface for managing some equipment. And I've played with them in years past. They have their appeal. They're typically small and lightweight. Now, we know that, until recently, just having been fixed, they've got some problems. And of course I know that this podcast's following has a way above average technical orientation. So I wouldn't be at all surprised if among our listeners there are people who are using VNC.

So mostly I wanted to make sure that everyone knew, because I don't know how tight the various VNC versions update communication is with its users. I mean, these things have been around for so long, they way predate any notion of auto updates. And maybe that's been fixed, or added later, or you're on a mailing list, whatever. But Shodan reveals more than 600,000 VNC servers currently accessible online. And Kaspersky believes the actual figure is likely to be significantly higher, probably because it's possible to get the servers to listen on nonstandard ports, and Shodan scans the expected port, verifies the protocol that is listening and answering, and says, oh, yeah, that's VNC. And so it adds it to its searchable database.

So the Kaspersky guys closely examined four common open source VNC implementations. LibVNC, as its name suggests, is a library of code on top of which developers can create fully functional customized apps. And LibVNC is used in systems that allow remote connections to virtual machines, as well as iOS and Android mobile devices. So it tends to be used. The problem is that, as is often the case with open source software, if the library gets updated, and this one needed updating, it's not always clear how those changes filter out into other applications that once imported that code base and may now have heavily customized it. So it could be that there are other applications using LibVNC that, whereas the library itself is going to get updated, it's not clear that all of the descendants of it will.

The second one was TightVNC 1.x, which it's the application recommended by vendors of industrial automation systems for connecting to their human machine interface systems. TurboVNC specializes in enabling remote work with high bandwidth things: graphics, 3D, and video objects. And then UltraVNC is the VNC variant which was specifically built for Windows, also widely used in industrial settings for connecting to human-machine interfaces. So across those offerings, one problem was found in TurboVNC; four were found in TightVNC; 10 in LibVNC; and by far the most in the Windows version, UltraVNC had 22 problems. And of course being a networked client and server solution, you need a VNC at each end, and the server will be listening.

The good news is what they found was the bulk of the problems were on the client side rather than on the server side. So that's good. They did note that server-side vulnerabilities were significantly less common than the client-side. On the other hand, the flaws that existed in the server side could allow authentication bypass, which of course we know in the now-infamous Windows BlueKeep RDP flaw is the reason it is such a problem. So they said that some of the attacks would be impossible without authorization.

So one of the things you absolutely want to do, if you've got VNC up and listening, is to make sure that you've got a strong password. We know that background automated password, so-called "credential stuffing" attacks are on the rise. So all the bugs involved incorrect memory usage which led to malfunctions and crashes. And as we know, that's where exploit work begins. If you can crash something with some sort of a buffer overflow memory use problem, the advantage that the attacker has is that all of this is open source. So they don't have to reverse engineer. They're not having to probe blindly at the server. If they see that they've been able to crash something and can identify what the server is, then they're able to go into the open source community, get the source, figure out what it is they did to make the thing crash, and see whether they could take advantage of it.

Anyway, so the good news is all of the developers were responsive except the creators of TightVNC, who said that since they no longer support the first version of their system, the 1.x branch, they're not patching the vulnerabilities which were found even though there is a substantial body of TightVNC v1.x on the Internet. So that might be any of our listeners. If you're using TightVNC 1.x, know that its updates have been abandoned, and that updates are known, and that some of them are critical. How many did I say were in TightVNC? Four in TightVNC, which the maintainers are now aware of, but have said they're not going to fix. The good news is they've moved on past the 1.x branch. So if possible, you want to move yourself over to an updated version of that.

So we can hope that the problems in LibVNC will get propagated out to any of the other projects which are using that library in order to take advantage of this virtual network computing technology, which is now really pretty mature. I took a look at it. I had some occasion to look at it a few months ago. And I was impressed that the things were still around. In general, we've seen a move over towards commercialization of these things. There were a lot of them that were rather feature lean until you signed up for the VNC as a service plan, which didn't fit the application that I had. But anyway, I've got the link to a detailed breakdown that Kaspersky provides of all of the vulnerabilities that were found.

And of course the standard advice, if you're going to run one of these things on the Internet, is if you can, if you always know you're going to be connecting from one or some low number of IPs, then filter. There's no reason to leave any server that doesn't need to be publicly accessible, to leave it publicly accessible. That is the number one thing I would recommend. For example, if you're always connecting from one office to another, and your office IP is not changing, just put up a firewall rule so that that port is only accessible from the known IP. It's a pain, if that IP does ever change. But it's a matter of just updating the firewall. And in the meantime, you're not open to the rest of the world. Period.

And you also want to make sure that you are running the latest version of VNC, if you know that you are a shop that has VNC in there. And as I said, make sure that you're using as strong a password as possible. These days, you know, nobody needs to type in a password. You can just cut and copy and paste. And I would say that's the kind of password that you want to be using in every instance.

So anyway, just a heads up. I'm just so glad that we've got an industry now where we're really beginning to take security seriously, and we've got so many security-related shops that are deciding, hey, let's go take a look at this and see what we can find. They responsibly disclose, these patches get applied to the software, and then hopefully this is the last link in the chain is that the word gets out that it's kind of important to update this software.

Leo: Remote access is hard because RDP has problems. There must be something about it. It's hard to do.

Steve: Yeah. Well, it is you know, in the case of RDP, and we sort of - we've touched on this a few times. Generally there's a protocol that is running over that connection. And a protocol probably means an interpreter. And, you know, one of our themes of the podcast is interpreting is difficult. You just, when you're writing the interpreter, you just assume that the data you're receiving is from the thing on the other end, the other automation, which produced the proper protocol. Because we had a hard enough time getting it to produce the proper protocol, so you're happy that it is. So your guard is down. You're not thinking of, like, oh, but what if somebody deliberately produced the wrong protocol? Well, you know, that tends to be the way all of these things happen, get exploited.

Leo: Yeah, yeah.

Steve: So Twitter finally allows SMS-based two-factor authentication to be disabled.

Leo: Turn it off.

Steve: Oh, my goodness. They announced last Thursday that its users will finally be able to disable SMS-based two-factor authentication for their accounts and use an alternative method only, such as the ones we like, the mobile so-called, you know, the one-time authenticator, the OTP app-based approaches. Or if you're, like, super secure oriented, a hardware security key. But until last Thursday, believe it or not, it was impossible. If users wanted to use any form of two-factor authentication for their Twitter account, they had to first register a phone number and enable SMS-based two-factor authentication, whether or not they wished to use SMS at all. And if you wanted to use a one-time password mobile authenticator app, or a hardware security key, you had to first enable SMS-based authentication in order to get that, and you could not disable it.

So the good news is apparently it took Jack Dorsey being hacked using a SIM swap attack. And we talked about this. I can't remember now what it was. I think it was when I was setting up my Hover account. I'd been at Network Solutions forever. I was moving all my domains over to Hover, which is now where I have all of my domains. And I was very happy to see that they offered two-factor authentication. And I remember coming on the podcast after that, specifically talking about why I was not using SMS-based messaging to second factor. I was delighted that Hover allowed me to use an authenticator app. And this was years ago. And of course now we've seen example after example of high-profile identification impersonation as a consequence of the fact that the telephone system, we've talked about how the Switching System 7, SS7 system, does not incorporate intersystem authentication. It doesn't even exist in the No. 7 protocol. It's just missing. It was never put in there.

So anyway, I just wanted to give our listeners a heads-up that you are now able to disable SMS-based second-factor authentication, keep one-time password authenticator app-based authentication, and you can also remove your phone number from your account, which also was not possible until recently. So yay. I'm not happy that Jack got himself compromised, but it certainly did bring to his and his security team's attention what we've all known for a long time. And I'm wondering, Leo, do you know any of the back story?

Leo: Oh, yeah.

Steve: Was it because tweeting started off as SMS?

Leo: Yes. So for a long time you could tweet via 40404, their short code. Until very recently you could. And there was a tool that would - I think it was a - I don't know if it was a Twitter tool. I think it was third-party tool, Twitter might have acquired them, that allowed you to tweet via your phone. Maybe you could turn it on. In any event, it didn't do any authentication. It just said, well, that's your phone number, so it must be you. And so the SIM swap meant that all they did was they got Jack Dorsey's SIM, they got his phone number, and then they could tweet as him. There was no further authentication after that, using 40404. So what they've really done is they've disabled 40404. So you can't - I don't think you can tweet via SMS anymore.

I'm a little confused by this story because I thought the same thing. In fact, I had experienced that, and I wanted to turn off SMS. For a while I couldn't. Somebody messaged me, said no, you can. What you have to do is, as you described, turn on SMS first, set up TOTP, then you could disable SMS. And I did that some months ago. So I'm not sure exactly [crosstalk]. Well, it may be that they hadn't ruled it out globally. Because I do remember a time when you couldn't do that. So maybe I was lucky enough. Maybe because I have a blue check. Maybe because they thought, oh, well, we've got to give this to some people right away. But I guess it's globally available now. And it's crazy that you couldn't do that, that you had to have an SMS verification. But that's not how Dorsey was attacked, to be clear.

Steve: Well, I mean, and tweeting was originally, as we know...

Leo: Was text. Well, sort of like text.

Steve: Well, it was 140 characters. It was because an SMS message was 160. And so you had 20 characters for the addressing, and then 140 for the message.

Leo: But that was the SIM dark age. That was 2006.

Steve: And so maybe it's just old...

Leo: 2007. I mean, but no, for a long time they kept 40404 around. In fact, it's in my address book as Mr. Tweet. So if I text Mr. Tweet, it would tweet it, yeah. And that's how they got Jack because, you know, there was no authentication on that system.

Steve: Yeah. Well, and we've often talked here about how often legacy stuff, which you're no longer using...

Leo: Oh, yeah.

Steve: ...but it still around, and it's still active, can come back and bite you in the butt.

Leo: I'm guessing some people still used it. That's why they didn't want to turn it off.

Steve: I think that's exactly right. It probably was like people with a feature phone who were like literally sending SMS messages in order to tweet.

Leo: Right. Yeah, because Twitter in a way was kind of an alternative text messaging service, really.

Steve: Yeah.

Leo: They finally arrested the guy who did it. One of the Chuckling Squad guys got arrested.

Steve: Ah. Good. Well, maybe that will cool off anybody else who wants to get up to such hijinks.

Leo: I hope so.

Steve: It's good to have some accountability. Speaking of accountability, I'm glad you know about the foregone conclusion. Jason and I talked about it when the foregone conclusion...

Leo: I'll tell you that story. This is a child pornographer. And they had his hard drive. There has been a court doctrine in the past that, if law enforcement knows what they're going to get when the safe is opened - we know the gun's in there, the guy has even said the gun's in there, but he won't give us the safe code - it's been okay to say no, you have to give us the code. It's a foregone conclusion what we're going to get.

Steve: Yes.

Leo: So it's not testimonial. The whole issue is testimonial. You're protected from self-incrimination by the Fifth Amendment of the Constitution.

Steve: Right. Yeah, so, and in the instance that Jason and I discussed, there was a woman in a car accident...

Leo: Oh, this is different. This is a different case.

Steve: ...who did not want to unlock her phone.

Leo: Oh.

Steve: And so the prosecutor said we know it's your phone because it was in your pocket. We know you use your phone because we have some records that demonstrated you were just using it. So it is a foregone conclusion that you are able to unlock your phone. Therefore you must do so. So it was...

Leo: No, I don't think that's it.

Steve: Well, no, [crosstalk].

Leo: It's not that we know you can unlock it, it's that the evidence on the phone is a foregone conclusion.

Steve: Yeah, well, in this case...

Leo: They know you can unlock your phone. They know you know your password.

Steve: Right. And so but that was, in the case that Jason and I discussed, that was, I mean, this was stretched in order to do that.

Leo: I see. This is not the case, however, that was just recently decided.

Steve: Correct. So now what has happened is there was an instance where the foregone conclusion was attempted to be used to compel the disclosure of a password. And so once again testimonial. That passed the lower court. But what just happened was that the Pennsylvania Supreme Court ruled that the Fifth Amendment to the U.S. Constitution, again, as we know, against self-incrimination, protecting someone from being compelled against incriminating themselves, did hold, and that the foregone conclusion approach that the prosecutors were taking did not hold.

And so it came down, though, it was a closely divided decision, a 4-3 ruling. They overturned the lower court order. And this was a child pornography case where a 64-character password was being used to protect the plaintiff's computer. And he said, essentially, he said no effing way I'm going to give it to you because we all know what's on the computer, and it's going to be bad for me.

Leo: That's the foregone conclusion right there.

Steve: Exactly. And so that was the logic that the prosecutors attempted to apply. It was appealed, and it was overturned on appeal. And I always like the carefully worded logic of these things, so I thought I would share exactly what the court's thinking was. They said: "Based upon these cases rendered by the United States Supreme Court regarding the scope of the Fifth Amendment, we [the Pennsylvania Supreme Court] conclude that compelling the disclosure of a password to a computer, that is, the act of production, is testimonial. Distilled to its essence, the revealing of a computer password is a verbal communication, not merely a physical act that would be non-testimonial in nature." And so maybe the unlocking the safe, maybe that's a physical act that's non-testimonial?

Leo: No, it's testimonial. That's the same thing. It's something in your brain.

Steve: Oh, okay, same thing.

Leo: So they can't demand the password. They can unlock the safe in other ways, just like, you know, they can physically open the safe.

Steve: Right, break it, right.

Leo: But they can't demand the contents of your brain. And I was looking at the Florida case you were talking about. You're absolutely right. The problem in the Florida case was they were trying to redefine "foregone conclusion" to be, well, we know she knows the passphrase. The court said no, no, no, that's not what it means. It means you have to know that once you get in the phone, there is the evidence you want. And you know it from other sources. And this is in the case of the child pornographer. They know what's on that hard drive. He's even said it. And he's even said, but I'm not going to let you see it because then you put me in jail.

Steve: Yes. Yes.

Leo: It is a foregone conclusion, but that's not sufficient. Anyway, continue. I'm sorry.

Steve: Yeah. So the court said: "There is no physical manifestation of a password, unlike a handwriting sample, blood draw, or a voice exemplar." They said: "As a passcode is necessarily memorized, one cannot reveal a passcode without revealing the contents of one's mind. Indeed, a password to a computer is, by its nature, intentionally personalized and so unique as to accomplish its intended purpose - keeping information contained therein confidential and insulated from discovery."

They said: "Here, under United States Supreme Court precedent, we find that the Commonwealth is seeking the electronic equivalent to a combination to a wall safe - the passcode to unlock Appellant's computer. The Commonwealth is seeking the password, not as an end, but as a pathway to the files being withheld. As such, the compelled production of the computer's password demands the recall of the contents of Appellant's mind, and the act of production carries with it the implied factual assertions that will be used to incriminate him. Thus we hold that compelling Appellant to reveal a password to a computer is testimonial in nature.

"We acknowledge that, at times, constitutional privileges are an impediment to the Commonwealth. Requiring the Commonwealth to do the heavy lifting, indeed, to shoulder the entire load, in building and bringing a criminal case without a defendant's assistance may be inconvenient and even difficult."

Leo: But it's right.

Steve: Yeah, yeah, exactly. "Yet to apply the foregone conclusion rationale in these circumstances would allow the exception to swallow the constitutional privilege. Nevertheless, this constitutional right is firmly grounded in the realization that the privilege, while sometimes a shelter to the guilty, is often a protection to the innocent. Moreover, there are serious questions about applying the foregone conclusion exception to information that manifests through the usage of one's mind."

Leo: This does not settle it, of course.

Steve: No.

Leo: It's just the Supreme Court of Pennsylvania, and there's many other courts. Part of it is - here, get this. Part of the foregone conclusion doctrine that they're trying to bend it into is, well, we know that a decrypted version of the hard drive exists. So, somewhere.

Steve: Somewhere. In theory, yeah.

Leo: In theory. So for that reason we should have access to it. I'm hoping, but again, this is up in the air, other courts have not held the same way, so far it looks like the trend is going in the right direction, that that is considered testimonial, and you can't testify against yourself.

Steve: Yeah, yeah.

Leo: Let's hope so.

Steve: And of course we've discussed this because the question is do I lock my phone with a password or a biometric? Because one is testimonial, and one is a physical manifestation.

Leo: Right.

Steve: You know, like handwriting recognition or something.

Leo: Yeah.

Steve: Yeah. So I wanted to reverse myself, Leo.

Leo: Uh-oh.

Steve: Because I do want to find out what happens to Baby Yoda.

Leo: Did you at some point say you didn't? You're going to get Disney+, is that what you're saying?

Steve: Well, the bill, I saw the email because a week had lapsed, and so my seven-day trial was up.

Leo: And you said you hated "The Mandalorian."

Steve: Well, yeah, I did. And I'm going to watch it.

Leo: Oh. See, I have still not paid for Disney+. I haven't even started the one week because I just don't know. So it's getting good now?

Steve: Well, it's not bad. And...

Leo: And Baby Yoda? Is Baby Yoda in it? Is that the thing?

Steve: Yeah.

Leo: Ah.

Steve: And but not THE Yoda. And that was, I mean, I heard some discussion at the beginning of MacBreak Weekly about this.

Leo: It's a meme now. "Boomer, okay." Yeah.

Steve: And so, you know, Yoda, I don't know if we ever knew the name of the creature. I mean, the species. Because Yoda was the name of that green guy with the ears; right?

Leo: I think he's a rubber puppet species, but I'm not sure. He seems very rubber puppet-like.

Steve: Well, I did - I think I'd only seen the first episode when I talked to you last week. I said, ohhh.

Leo: Right. You couldn't even get through it, I think, yeah.

Steve: Barely made it. But then little Baby Yoda manifested, and I thought, okay. I'm, you know. And Lorrie looks at me, "Really? You're going to make me watch this?" I said, "No, honey, you can read or something." But I've got to find out what happens.

Leo: I respect your right to do that. My opinion changed of the Apple TV Plus "Morning Show." I hated the first couple of episodes, and it started to grow on me. I think it's kind of, honestly, a Stockholm Syndrome thing, that after a while, if a TV show's especially bad, you've spent so much time, you've invested so much time in it, that you start to love it. You just say, well, I'm going to keep watching because it's...

Steve: Okay, now, "The Leftovers" was like that for me.

Leo: Oh, god, I hated the freaking "Leftovers."

Steve: Oh, my god.

Leo: And I hate-watched that one. You're right.

Steve: We were told that it was, by the end, it was the best television ever made.

Leo: No. Not even close.

Steve: By somebody whose opinion I sort of used to respect. And so Lorrie and I kept waiting for it to get good.

Leo: Never gets good.

Steve: For three seasons.

Leo: Yes. Oh, you watched more than I did.

Steve: And oh, my god.

Leo: Yeah. "The Affair," same thing. I thought, this should get good. These are good actors. This should be a good - this should be better.

Steve: Oh, and it's well regarded, too.

Leo: And it's highly reviewed.

Steve: Yeah.

Leo: Yeah. And somehow this affair's gone on for four years, four seasons. I don't know. You know what, there's so much good TV, Steve, you don't need to watch the stuff that's only mediocre.

Steve: Well, in fact I heard you mention "The Crown." And that's...

Leo: Oh, that's a good show.

Steve: Yeah? Good.

Leo: Do you like that stuff?

Steve: I don't know. I'm going to try. Lorrie will definitely.

Leo: She'll like it, yeah.

Steve: And so I'll...

Leo: Did you watch "Downton Abbey"? Did you like "Downton Abbey"?

Steve: No, didn't try that.

Leo: Yeah. See, I think these are more for Lorrie than you. It's funny because in our house I'm the girl, and I love...

Steve: I heard about you crying at the end of the Apple commercial. So, yeah.

Leo: Lisa says, "Are you crying at a commercial?" And I said, "Yes."

Steve: And actually I get what it was that had to have happened, even though.

Leo: Oh.

Steve: And so I can understand. I can understand how moving it would be.

Leo: It was beautiful. It was beautiful.

Steve: Now we have to apologize to our listeners for the last 10 minutes of their lives.

Leo: I'm so sorry, wasted. That's all right, you can hate-listen to this, too. No, no.

Steve: Okay. So I'm going to do the first half of Pushing DoH, which doesn't give - which is sort of a good introduction. It's the reason I got into this, which is that Microsoft has not only just jumped into the DNS-over-HTTPS world, but has in one fell swoop instantly legitimized the various web browsers' efforts to protect their users' DNS queries from the prying eyes of their ISPs. So sorry, Concast.

One of the other things that Jason and I talked about, Leo, was that Comcast put together a really, it should be embarrassing, set of arguments to Congress about Google.

They singled out Google because of course Google's the whipping boy du jour, hiding DNS queries from them, subverting the Internet. Well, and of course we talked about how the U.K., what was it they called Mozilla?

Leo: Enemy of the people or something, yeah.

Steve: It was awful, yeah. And then they backtracked on that after they got a lot of pushback. It seems clear that we are on the brink of evolving DNS away from UDP, if only, and apparently only, because remember this only is the first hop of DNS. It's not DNSSEC where it absolutely protects it from hacking. And it's not the deep DNS which will still be moving across the Internet via UDP. It's just keeping your ISP's hands off of it is all this really does because it's only creating an encrypted tunnel from your browser to wherever you have pointed your DoH client to get DoH resolution. But what's interesting is Microsoft has formally stated that they're going to be adding DoH to Win10, to Windows 10. It's going to get native support.

Here's what they said. They said: "Here in Windows Core Networking we're interested in keeping your traffic as private as possible, as well as fast and reliable." This is Microsoft speaking. "While there are many ways we can and do approach user privacy on the wire, today we'd like to talk about encrypted DNS. Why? Basically, because supporting encrypted DNS queries in Windows will close one of the last remaining plaintext domain name transmissions in common web traffic.

"Providing encrypted DNS support without breaking existing Windows device admin configuration won't be easy. However, at Microsoft we believe that 'we have to treat privacy as a human right. We have to have end-to-end cybersecurity built into technology.' We also believe," they said, "Windows adoption of encrypted DNS" - that is, Windows' own native adoption of encrypted DNS - "will help make the overall Internet ecosystem healthier. There's an assumption by many that DNS encryption requires DNS centralization. This is only true if encrypted DNS adoption is not universal. To keep the DNS decentralized, it will be important for client operating systems (such as Windows) and Internet service providers alike to widely adopt encrypted DNS.

"With the decision made to build support for encrypted DNS, the next step is to figure out what kind of DNS encryption Windows will support, and how it will be configured. Here are our team's guiding principles on making those decisions." There are four of them. First: "Windows DNS needs to be as private and functional as possible by default without the need for user or admin configuration because Windows DNS traffic represents a snapshot of the user's browsing history. To Windows users, this means their experience will be made as private as possible by Windows out of the box. For Microsoft, this means we will look for opportunities to encrypt Windows DNS traffic without changing the configured DNS resolvers set by users and system administrators."

Two: "Privacy-minded Windows users and administrators need to be guided to DNS settings even if they don't know what DNS is yet. Many users are interested in controlling their privacy and go looking for privacy-centric settings such as app permissions to camera and location, but may not be aware of or know about DNS settings or understand why they matter, and may not look for them in the device settings." In other words, yeah, people just don't think about DNS as being a privacy issue. They don't even know what it is, often; it's so behind the scenes.

Three: "Windows users and administrators need to be able to improve their DNS configuration with as few simple actions as possible. We must ensure we don't require specialized knowledge or effort on the part of Windows users to benefit from encrypted DNS. Enterprise policies and UI actions both should be something you only have to do once rather than need to maintain." And, finally: "Windows users and administrators need to explicitly allow fallback from encrypted DNS once configured. Once Windows has been configured to use encrypted DNS, if it gets no other instructions from Windows users or administrators, it should assume fallback to unencrypted DNS is forbidden."

So anyway, what's interesting to me is that they're not doing this soon. They're just saying we're making a statement that we concur with the needs to encrypt DNS. So they said: "Based on these principles, we're making plans to adopt DNS over HTTPS (or DoH) in the Windows DNS client. As a platform, Windows Core Networking seeks to enable users to use whatever protocols they need, so we're open to having other options such as DNS over TLS in the future. For now, we're prioritizing DoH support as the most likely to provide immediate value to everyone." Well, and of course the reason is the browsers are tending to drive servers to do DoH, so those are the servers that'll be available for Windows to natively talk to, so DoH it is.

They said: "For example, DoH allows us to reuse our existing HTTPS infrastructure." And that actually is the trick that I'll be talking about in a minute. They said: "For our first milestone, we'll start with a simple change. Use DoH for DNS servers Windows is already configured to use. There are now several public DNS servers that support DoH; and, if a Windows user or device admin configures one of them today, Windows will just use classic DNS, without encryption, to that server. However, since these servers and their DoH configurations are well known, Windows can automatically upgrade to DoH while using the same server. We feel this milestone has the following benefits."

And I'm going to skip the enumeration because we know what they are. That is to say, if you have configured Windows to use some whatever it is, DNS servers, Windows in the future will test them to see whether they support DoH and, if so, will automatically preferentially switch to DoH rather than traditional DNS over UDP.

So they said: "In future milestones, we'll need to create more privacy-friendly ways for our users to discover their DNS settings in Windows as well as to make those settings DoH-aware." Meaning, for example, a user would have the ability to say I'm pointing you at a DoH DNS resolver. I only want DoH. Do not fall back if something is blocked to DNS over UDP. And of course we know about downgrade attacks where it would be possible to block DoH queries to a configured server in order to cause the person to use DNS, and then you could play DNS games with them.

So they said: "Why announce our intentions in advance of DoH being available to Windows Insiders? With encrypted DNS gaining more attention, we felt it was important to make our intentions clear as early as possible. We don't want our customers wondering if their trusted platform will adopt modern privacy standards or not." Okay. Which seems odd to me. I have to believe that some sort of lobbying at the political level was going on behind the scenes with the browser vendors. I mean, for one thing, Microsoft is now all Chromium for Edge. That's where they're moving. Chromium does support DoH. So presumably Edge, well, actually we know that Edge does because we talked about that a couple weeks ago, that you could already turn DoH support on in every major browser except Safari. Opera has it. Chrome has it. Firefox has it. Vivaldi has it. And of course Vivaldi is Chromium based, so that's where it got it. So all the browsers can support that currently.

And I just look at this very strong letter that Comcast wrote to Congress, and the fact that Mozilla then rebutted it, just like took it apart point by point. And then here's Microsoft saying, you know, we believe in DoH. We're building it in. Yet you can't get it today. I just sort of think, okay, they were sending a signal maybe that will be useful to the rest of the industry, that this looks like the direction we want to go.

So it turns out that there's a bit more to DoH than may at first be obvious. And I wanted to share what I learned after doing some additional digging. So the first thing to observe is that our goal is encryption to get privacy and certificate verification to know who we're connected to for authentication. But that means that we have both DoH, DNS-over-HTTPS, and DoT, which you heard Microsoft refer to. That's DNS-over-TLS. Either one would give us encryption for privacy and certificate verification for authentication. And since DoH is DNS-over-HTTPS, and HTTPS is HTTP-over-TLS, my immediate instinct was to prefer the simpler of the two solutions, which would simply be DNS-over-TLS with no HTTPS involvement at all. So I've been sort of wondering, okay, why DoH? DoT is simpler, leaner, cleaner, and so forth.

Well, it turns out that DNS could very nicely leverage some of the advances we've previously discussed in the HTTP protocol as we moved from HTTP 1.1, where we've been forever, to HTTP/2. And we did a podcast about HTTP/2 (SN-495), all the new features that it is bringing to us. And I'll be referring to some of those, so you don't have to go back. But okay.

So in the first place, rather than create a new URI label such as dns://, you know, in the same way we have, like, mailto and ftp and the other URI labels. The designers of DoH chose to reuse all of the external HTTP protocol, so it's just https://, and instead to identify DNS queries using standard HTTP query headers. So the path of the query would, for example, be "/dns-query." So that would be the root of the query would be /dns-query. And then the accept types header, which indicates to the server what type of material you're willing to receive in response to the query. There's a new application type, application/dns-message. And the content type header would also be, and that's essentially the format of what it is that you're going to be receiving would be similar, application/dns-message.

So these headers clearly identify and flag the query as, even though it's going over HTTPS, as being a DNS query and wanting a DNS reply. And it turns out that both the GET and the POST-style queries are supported. They both carry, in terms of the format of the query, by default the exact same binary identical query data on the wire as DNS itself uses. So the payload of a DNS UDP packet is, for example, in a POST query, the body of the post would be the binary identical DNS query. And in the case of a GET, that binary DNS query data would be base URL encoded to turn it into ASCII so that it could be sent as the query tail of a GET query. And in both cases the replies to these queries are exactly the same binary DNS data as would have been returned over the wire via traditional UDP query.

So, you know, we know that HTTP is already capable of bringing binary data. That's what our various image formats are, and arguably even the compressed texts that we get in Gzip compression. That's non-ASCII, it's binary, so there's no problem receiving the result from the HTTP tunneled query in binary. There is one little variation on this, which is the IETF, which is working to standardize the use of DNS on the wire format, does allow JSON encoding. So the accept header can specify application/dns-json, J-S-O-N, in which case there is an RFC 8427 which standardizes the JSON format if for some reason you want a pure textual, both a transmission and a reception of DNS using JSON instead of binary.

Okay. So the point is that in every way, DoH is a DNS query that fits entirely within HTTP and is completely compatible with HTTP. The fact that you're getting binary is fine with HTTP. All of our images come over the line that way. And it's only the query headers which indicate to the servers that that's what this is. This happens to be a DNS query rather than give me the next page or give me some content, give me an MP3 or whatever.

So the way this was defined, DNS becomes just another HTTP query. So we've often talked about the crazy mess that a typical website or a web page has become, where the browser pulls down and parses the base page, which to a crazy degree these days then refers to scores of other domains from which it's pulling ads, it's pulling JavaScript scripts from every direction, I mean, it's just this incredible conglomeration of assets which the browser needs to aggregate in order to pull this page together.

And of course so it gets the first page. It parses the page. It finds all of the URLs. It finds all the domain names for the URLs. Now it has to go and generate a blast of DNS queries in order to turn the domain names into IP addresses, wait for all the queries to come back to get the IP addresses, then initiate connections out to all of those servers in order to ask them for whatever asset it is that the original source page it downloaded has told the browser it needs to get.

So that's a lot of work. And if we've seen anything over the past decade of work on the web, it's been web browsers and their protocols being revised and updated and massaged to provide the fastest possible experience for their visitors. Google has spearheaded a bunch of improvements which were initially experimental, and then they worked, and now they've moved themselves into, sometimes merged and sometimes whole, into mainstream web browsing protocol. So as I've said before, hats off to Google for pushing a lot of these innovations.

We talked about HTTP/2 is inherently a multiplex protocol, whereas the original HTTP was a single request and a query, after which the server would disconnect. Then we said, okay, that's dumb because now we have to connect again if we want to ask the same server for something else. So for a while the browser was setting up lots of connections in parallel. Then we switched it around to maintaining a persistent connection. And so after the reply to the first query came back, it was possible to ask the connection to be kept alive so that the browser could then send another query over the same connection and get a second thing back.

HTTP/2 dramatically changed that so that now we have essentially a meta connection where we establish a connection, and then we have a multiplex protocol where individual queries are numbered and tagged and can all be sent out in a blast. And the data that is coming back, the answers to the queries are in packets which are tagged, which allows the browser to essentially simultaneously receive the results of its queries in any order that the server wants to send them. And then the browser reassembles them on its end.

And as we've discussed, HTTP/2 even allows for the web server to anticipate the content that it knows the web browsing client will eventually be needing even before it asks for it. After all, it's sending the page. It knows all of the assets which eventually the web page is going to need in order to be displayed. So why send it and sit there idly waiting for the client to read the page, parse the page, and start asking for things? It might as well use that time by just sending ahead, pushing the content that it knows the web browser client will eventually be needing. And of course it does that.

And if anybody is interested and is confused by any of this HTTP/2 stuff, we did talk about it in the past. And all of our podcasts are available, as we know, online historically. So you could go back and brush up on HTTP/2 if you were curious. And so what happens is the browser will simply find the asset already in its cache, as if it had been pushed by the server to the browser ahead of time.

Okay. So with that background, imagine a future where the HTTPS website that's being connected to also offers DNS resolution services via HTTPS. In other words, the website that is offering web pages is also a resolving, a recursive DoH server. It's already serving HTTPS. It's already got a certificate. You've already established a connection. DoH is just an HTTP query. So it completely folds into all of the existing protocol that the client and server are operating on, including the benefits of HTTP/2.

So nothing has to change. In this future, the website knows all of the large number of domains being referenced by the page the browser has requested. And DNS is inherently a caching protocol and system. So the DNS resolving web server can have previously looked up and cached the most recent IP addresses for all the domains it will be asking the browser to go and fetch assets from. And using HTTP/2's built-in send-ahead push technology - thus the title of this podcast, "Pushing DoH" - that web server could prepopulate the client's local DNS cache with fresh and valid DNS lookup addresses, thus completely short-circuiting all of the time required for the client to do DNS lookups for all the domain names in the URLs which appear on the page.

Now, yes, for this to be safe we would probably want the website's DNS replies to be DNSSEC. So when DNSSEC matures and becomes available, the replies can be verified secure coming from the resolving site. On the other hand, think about it. We're already trusting the web server we're connecting to, to provide us with a whole bunch of content that our web browser is going to go and get. I mean, it's already providing the URLs containing the domain names that we go look up. So it's not clear that there's a lot of danger inherent in accepting the IP address for the domain that we were going to look up and get an IP address for anyway from the web server. But when you think about it, this change of model is so much more efficient overall.

In the original, completely distributed DNS model, every client of an active website which is going to get web pages is needing to perform its own DNS lookups. And, like so for everybody that gets that page, they're having to do DNS lookups for all of the assets on that page. And yes, it's broadly distributed. That's what DNS is. But why not have the web server do the DNS lookups and then pipe those ahead to the client while the client is receiving the web page and figuring out what it is that it's got to do so that it already has the DNS lookups done ahead of time.

And this is not fantasy. It turns out these are the plans for where DoH is headed in the future. And that explains why DoH is what people have jumped on, rather than DoT, DNS-over-TLS. Even though that's a lot leaner, it doesn't have the advantage of being able to share the existing connection, which already exists, and take advantage of all the advances that we've made in HTTP in order to create a multiplex protocol. And what this potentially does is dramatically improve the speed at which web pages are able to load. So, way cool. Way cool.

Leo: Nice. Nice side effect, yeah.

Steve: Yeah. Yeah.

Leo: Very nice.

Steve: And the user does nothing. We do nothing. The web browser will be able to check, generate a query, see if a DoH query is handled. Or it'll just be receiving DoH replies for lookups it hasn't even gotten around to asking for yet. And those will then contain non-expired DNS queries which the web server that it connected to will have recently looked up and cached. And of course DNS, as I said, is a caching protocol. It's got expiration times and timeouts. So it just beautifully flows. And it's just very, very cool.

Leo: Sometimes doing the right thing has a benefit. Nice. You might do it for one reason and get some of the other effects. So I guess DoH is coming. It's too bad that DNSSEC did not take off. Then we wouldn't need this; right?

Steve: Well, no. DNSSEC provides protection for the records, but no privacy.

Leo: Oh.

Steve: So DNSSEC is not an encrypted protocol. It's a signature protocol. So it signs the records and prevents them from being forged.

Leo: Forged, okay.

Steve: But it does not give you privacy.

Leo: Interesting.

Steve: But we still need this in order to not be eavesdropped on. And that, boy, that really does seem to be the main driver for this is that - and it's interesting, too, that ISPs are kicking up such a fuss. They really don't want to lose that information.

Leo: Is there any logical other reason like, well, but then we can't optimize or something? I mean, is there any other, I mean, is there any conceivable justification for them saying that we want this?

Steve: I just think, I mean, so they can recapture the information by themselves supporting DoH.

Leo: Right.

Steve: So, I mean, it's not like - it's not a mystery. If they wanted, you know, right now they're doing...

Leo: Good point. That's actually a good point.

Steve: Yeah. Right now they're doing the DNS for their clients, typically, unless the client has manually overridden DNS.

Leo: Right.

Steve: So all they have to do is just bring up, I mean - and if DoH happens, then all the DNS servers are going to offer it anyway.

Leo: Yeah. That's the one problem right now. At least with Firefox's you only have Cloudflare.

Steve: It's very - yes.

Leo: And really, if you're not - do you trust Cloudflare over your ISP?

Steve: Well, Cloudflare, Google...

Leo: Google now has one.

Steve: OpenDNS offers it.

Leo: Oh, okay. But the default in Firefox, there's only one choice. That's Cloudflare.

Steve: That's true.

Leo: You have to manually enter in - you'd have to find and enter in the address for the others.

Steve: That is true. Although the reason it's only been Cloudflare is that Mozilla...

Leo: Trusts them.

Steve: Really, well, trusts them, and I think we do, too.

Leo: Yeah, right.

Steve: And we can, true. And they extracted a very powerful pledge about not keeping any query data beyond the time necessary to service the request. So it's very much like the ExpressVPN guys who are absolutely not doing any logging. They're like, we don't want to have the information.

Leo: Right. And I imagine Cloudflare uses the transit information to help optimize their own Cloudflare services. So there's a benefit to Cloudflare. They're not doing it completely for nothing.

Steve: Yeah.

Leo: But not an invasion of privacy, and that's the key.

Steve: Right.

Leo: Yeah. Thank you, Steve. Hope you enjoy this show. And if you do, I hope you'll make a point of coming by every Tuesday around about 1:30 Pacific, 4:30 Eastern time, 21:30 UTC. That's the easy one, Leo. 21:30 UTC. We do the live show on a stream so you can watch or listen live at TWiT.tv/live. You're basically watching us make the show. Of course then the editors get to it, clean it up, polish it up, put some stuff at the beginning and the end, and put it up on the website. Steve has a 16Kb audio version of it, as well as a 64Kb audio version, at his site, GRC.com. He also has something unique: transcripts. So if you want the transcripts, GRC.com.

While you're there, take a look at SpinRite, the world's best hard drive maintenance and recovery utility. Get it now, and you'll be getting 6.1 the minute it's ready. You'll also be able to use lots of free services - Steve's very generous - ShieldsUP! and so forth. And of course you can find out more about SQRL, GRC.com. He's on the Twitter at @SGgrc, and that's a good way to reach him. You can direct message him, anybody can, @SGgrc. We have audio and video of the show. If for some reason you want to watch it, you can at TWiT.tv/sn for Security Now!.

I do encourage everybody lately on the shows to subscribe. There's a couple of advantages. For you, the advantage is you don't have to think about it. You'll just have it the minute it's available. That's the key to a podcast. But the advantage to us is, by subscribing in your favorite podcast app - and I don't care which one you use, Apple iTunes or Stitcher or Slack or Overcast or Pocket Casts, whatever, Podcast Republic, whatever you use. Subscribing tells them, oh, people listen to this show, and maybe they'll feature us from time to time, which we would sure like. That helps us with discovery. So subscribe; okay? Awesome. Thanks for being here. And we'll see you next Tuesday on Security Now!. Bye, Steve.

Steve: Bye, Leo.