Credential Delegation

Description: This week we check in on the developments of the long-term, now working, full consumer jailbreak of iOS devices from the iPhone 4S through the iPhone X. We examine the strange case of the misbehaving transducer, catch up on the rapidly evolving exploitation of the BlueKeep vulnerability, check out Mozilla's rebuttal to Comcast's attack on DoH, examine the surprising state of web browser support for DoH, and remind Linux and BSD users to refresh their distros after an important flaw was disclosed in a widely used archive library. Then we take a deep dive into the operation of a newly announced forthcoming solution and standard for significantly improving TLS website certificate security known as "TLS Credential Delegation."

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-740.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-740-lq.mp3

SHOW TEASE: Coming up on Security Now! with Steve Gibson, I'm Jason Howell, my final week until Leo gets back next week. Steve's going to deep on TLS Credential Delegation. Also there's a number of BlueKeep updates that Steve reveals, as well as some good progress on browser support of DoH. And Checkra1n is now in beta, so you can check it out for yourself. Steve Gibson breaks all that down, next on Security Now!.

JASON HOWELL: This is Security Now! with Steve Gibson, Episode 740, recorded Tuesday, November 12th, 2019: Credential Delegation.

It's time for Security Now! with Steve Gibson, Jason Howell filling in for Leo one last week in a row. Leo gets back in a couple of days, Steve. So next week you're going to be doing this show with Leo in the other room. How's it going today?

Steve Gibson: Who?

JASON: Yeah, exactly, I know. That's what I'm saying.

Steve: Huh? Who? What? So, well, Leo has delegated the podcast the last four weeks to you.

JASON: Yes. I've been honored.

Steve: And today's topic is Credential Delegation for Security Now!.

JASON: I see what you did there.

Steve: Yeah, huh. Number 740 for November 12th. This is interesting. This jumps onto a - this is sort of an extension of one of my own long-running issues with secure connections. All of our longtime listeners know how wound up I have gotten in the past about the failure of TLS, or back then SSL, certificate revocation. So there's a new game that is beginning to emerge. Actually it's already in Firefox, which is cool. By the end of the podcast, our listeners are going to know how to turn that on and test it in their copies of Firefox.

So a lot of stuff happened this week. We're going to check in on the developments of the long-term, now working, full consumer jailbreak of iOS devices from iPhone 4S through iPhone 10. It's in its early beta, but it exists. We're going to examine the strange case of the misbehaving transducer. Catch up on the rapidly evolving exploitation of the BlueKeep vulnerability. There's been a lot of news there in the past week. We're going to check out Mozilla's rebuttal to Comcast's attack on DoH, examine the surprising state of web browser support for DoH. I didn't realize they all have it already. And I just stepped on the lead. Remind Linux and BSD users to refresh their distros after an important flaw was discovered in the widely used archive library which allows remote code execution.

Then we're going to take a deep dive into, as I said at the top, the operation of a newly announced forthcoming solution and standard, it's in the IETF's hands now, for significantly improving the state of TLS website certificate security for a subset of all use cases. This doesn't solve it for everybody, but it solves a class of a big problem known as TLS Credential Delegation. So I think another great podcast for our listeners at 740.

JASON: No surprise. 740, you've got this down. You've finally got it down, Steve.

Steve: Yeah, figured out how to do this.

JASON: We were doing the math before the show, going like, 2007? Was podcast even a term when this show launched? Like you've been around a long time with this show. And there's a reason why. You pack in the knowledge. So we are going to dive right into everything Steve said up there.

All right. CheckM8 is official. Like everyone? Legit with a capital "L"?

Steve: Well, not quite everyone. I don't know if it's ever going to be legit. It might be sort of like the anti of legit.

JASON: Lowercase "l," then.

Steve: But our Picture of the Week is a picture of the web page that we showed several weeks ago when we were cautioning our listeners about how to type in the domain name so they didn't go to Checkrain dot com by mistake and download some bad ware. But what we're seeing now, our Picture of the Week, down there is a link that says "Get the beta now."

JASON: Nice.

Steve: So we have moved into the first - actually, well, it's the first public betas because I checked again this morning. We have three of them so far. There was first v0.9, then there was 0.9.1, and now we're at 0.9.2. Oh, and also it's worth noting that the official site is now Checkra, C-H-E-C-K-R-A dot I-N. So that's the way they got - they got the Checkra in the .in top-level domain. So that's where they're aiming people. And if you were to go to the original Checkra1n where the "I" is a numeral "1" dot com, it just bounces you over to the proper location.

JASON: That makes way more sense.

Steve: So as we know, this is the public preview of the USB tethering iOS jailbreak, which is based on the unpatchable iOS CheckM8 bootrom exploit which can never be fixed, which Apple can never go back and fix on all the devices up until they found this last summer. So up to 12 point - I guess up to and including, I think, because now I'm confused, because I did see some - when I was digging into this there was some comment about 12.3.2 and 12.3.3. But maybe I'm misremembering it, and it was 12.2. But anyway, it's the devices based on the A5 through the A11 chip, but not the A12 and 13. And not yet everything.

So, for example, this 0.9.2 beta explains, it says: "This release is an early beta preview, and as such should not be installed on a primary device. We strongly recommend proceeding with caution." Then under "What's new," to give our users a sense for what's going on here, they said: "Fixed an issue where the Apple Watch would not receive notifications while jailbroken. Improve reliability of entering DFU mode. Fixed an issue where Checkra1n could not be used on macOS 10.10." Oh, it's worth noting that right now it's strictly a Mac-based launch of this. They'll be getting to other stuff, to Windows and Linux, subsequently. But at the moment you have to have a Mac in order to do this, this early beta preview.

Then they said: "This beta adds an option to boot into no-substrate mode. To utilize this functionality, hold the volume button up when the Apple logo appears until the device finishes booting. From there you'll be able to uninstall any tweaks causing you issues and reboot to get back into a normal jailbroken state."

Then, and here's the most important part at the moment, unsupported devices. They said: "Checkra1n will eventually support all devices between the iPhone 5S and the iPhone X; however, this beta lacks support for the following devices: iPad Air 2, iPad 5th Gen, iPad Pro 1st Gen." And then they said: "Support for these devices will be added in a later release. Support for the following devices is experimental and may require more attempts than usual: iPhone 5S, iPad Mini 2, iPad Mini 3, iPad Air." They said: "Reliability on these devices will be improved in future releases." So there are some where they're saying we got it; there are some where they're saying, yeah, we almost got it; and then there are some where, like, no, don't even try this yet. We'll get you. And they said: "This beta only is available for macOS. Work is ongoing to support Windows and Linux, which will be added in a later release."

Anyway, so just an update. This is, I mean, for the security industry, this is just a windfall. Apple has to keep the more recent of those devices current with all of their new patches and fixes. This means that researchers, both wearing white hats and black hats, will be able to crack into any of these iOS devices which are supported in order to deeply, I mean, like completely, thoroughly reverse engineer what Apple has done.

So, you know, this is like the worst thing to happen for Apple's attempt to keep people out of what they've done. We know that security through obscurity is not something you can rely on. But if you can get obscurity in addition to security, why not get it? Because reverse engineering does help bad guys discover problems that have not otherwise been discovered. And this makes iOS, until at some far future point, the iPhone 10 goes out of update cycle, it makes it deeply reverse engineerable.

So the expectation is, I mean, this doesn't represent any kind of a security problem for your typical end user, except if someone were to take your phone into the back room, use this to jailbreak it, and then you were unaware that that had happened, it would be a way for border agents entering a hostile country to get deep access into an iOS device if you were unaware. So certainly this says, because this cannot survive a restart, that if you are traveling into a country where you have lost control of your iOS devices even

briefly, it's worth completely powering down and restarting it clean because what this has done is it has made those devices always jailbreakable with a USB connection to the device in order to make this happen. And it's open. It's going to be widespread. I mean, again, this has been a real hit to Apple security, not something they're happy about, but not something they have any control over.

JASON: Yeah, that was my follow-up question is what could they really do in this? They've been pretty good at kind of keeping things like this at bay. But this really just sounds like such a - blowing the lid wide open, you know.

Steve: Yeah. Jailbreaking had been, I mean, it's the whole concept of secure boot, where the ROM verifies the signature of the kernel before it loads it into RAM. Then it transfers control of the kernel. The kernel verifies the signature of all the other stuff that it's loading piece by piece. And nobody but Apple is able to sign any of this. Well, that's gone now. I mean, so Apple was able to use the secure boot chain to absolutely guarantee to its users, and we like that, that when we start our phone up - and the stuff that's running in our phone has been signed and has been verified all the way through startup. And then we know that any apps that we are running are having their signatures verified also. That's the part that breaks, which then allows - means that malware can be installed, can be implanted on our iOS devices by anybody who has access to the device.

So Apple doesn't like this, I mean, doesn't like that compromise of their security guarantee that until now they've been able to offer all of their users. Nor do they like the fact that everybody, security researchers, can now easily sift through their code looking for as-yet-unfixed problems. And we know there are as-yet-unfixed problems because Apple is constantly fixing problems which were unknown until they fixed them. And in fact this also means that, in the same way that Windows patches have been, are being immediately reverse engineered and turned into relatively short-lived exploits, so too can Apple patches be. When Apple fixes something, since Apple rather lazily rolls out these iOS updates, somebody can get it, figure out what changed, reverse engineer it, find the problems. If it's useful to a bad guy, they can then exploit it until such time as the targets get themselves updated to newer versions of iOS.

So, yeah. And it was only recently that we got the stronger iOS auto update feature in our iOSes. So the older ones of iOS still covered by working devices may not be updated as quickly as the newer ones. So, yeah. And here it is. It's gone from CheckM8 to now we actually have what we were expecting, which is the first working exploit, which anybody can download. And the guys who are doing this are having a ball. They're going to, just because they can, they're going to make it robust and bulletproof and reliable and working on all of the devices within the range. So, very cool.

JASON: Very interesting, yeah.

Steve: Very, very, yes. Maybe not so cool as it is interesting.

JASON: Yeah, yeah. I'm really intrigued by this next story, as well, because we talked about this a little bit last week on This Week in Google. And I'm curious to get your take on this.

Steve: You know, I saw that news also before last week's podcast, and I thought, eh. But upon digging into it more, I thought, okay, this is kind of fun. In one of the weirder pieces of research - and I titled this "The Case of the Misbehaving Transducer." In one of the weirder cases of recent research, from a researcher with Japan's University of Electro-Communication - there is such a place - and four researchers with the University of Michigan, we have their paper titled "Light Commands: Laser-Based Audio Injection

Attacks on Voice-Controllable Systems." It's like, wait a minute. Laser-based audio injection? Huh?

So, okay. By definition, a transducer is any device that converts energy from one form to another. That's the dictionary definition of a transducer. Typically, it converts a signal in one form of energy to a signal in another. But it doesn't have to. But, for example, in the case of a signal, we all know that a speaker is a transducer that converts electrical current into movement of a cone, which then reproduces sound. And going in the opposite direction, a microphone is a transducer that converts incoming sound into movement and then into electricity. An example of a non-signal transducer is a light bulb, which converts electric current into light; and solar panels, going the opposite direction, converting light back into electric current. So transducers. One form of energy into another.

And in the past we have encountered some weird transduction behavior such as the famous case of shouting at an array of hard drives, which increased their read error rates and reduced their measured data throughput. That way we had fun. There was like a YouTube video of some guy who looked like he'd lost his mind, screaming at a RAID array. But sure enough, it upset the hard drives. They were functioning as audio transducers, even though they were certainly never meant to be that. There was another instance where a whole datacenter crashed because the incredible loud fire siren whistle thing went off, and hard drives all failed in there because this thing was so loud, and hard drives have become so twitchy about any mechanical perturbation because their density's gotten so high that they've just, you know, the engineers have engineered out all the margin for error.

So now we have another oddball transducer failure. As they describe it in their 17-page paper, they said: "We have identified [what they described as] a semantic gap between the physics and specifications of MEMS" - which is the acronym for micro-electro-mechanical systems - "microphones, where such microphones unintentionally respond to light as if it were sound." They said: "Exploiting this effect, we can inject sound into microphones by modulating the amplitude of a laser light." In other words, these microphones are light-sensitive as a weird side effect. Who woulda thunk? And you've got to wonder who even thought of trying this. Let's aim a laser at our Amazon Echo and see what happens. What? So, okay. So think about this. Someone located hundreds of meters away could tell your Amazon or your Google voice assistant to unlock your front door or open your garage door. That's not good.

So in their paper's Summary Abstract, they wrote: "We propose a new class of signal injection attacks on microphones based on the photoacoustic effect, converting light to sound using a microphone. We show how an attacker can inject arbitrary audio signals to the target microphone by aiming an amplitude-modulated light at the microphone's aperture. We then proceed to show how this effect leads to a remote voice-command injection attack on voice-controllable systems. Examining various products that use Amazon's [Echo], Apple's [you know who], Facebook's Portal, and Google Assistant, we show how to use light to obtain full control over these devices at distances up to 110 meters and from two separate buildings.

"Next, we show that user authentication on these devices is often lacking or nonexistent, allowing the attacker to use light-injected voice commands to unlock the target's smart lock-protected front doors; open garage doors; shop on ecommerce websites at the target's expense; or even locate, unlock, and start various vehicles" - and their examples, a Tesla and a Ford - "that are connected to the target's Google account. Finally, we conclude with possible software and hardware defenses against our attacks."

Anyway, I'm not going to go into this in any greater detail since everyone gets the idea here. But if we were to re-express this in more classic security terms, to understand why

there's this problem, we would say that the lack of authentication that's present in today's voice command systems represents a rational tradeoff between convenience and security when used in the typical environment where it's not possible for a random stranger at a distance to inject their own authenticated voice commands into those devices.

So this lack of authentication is obviously not normally a problem, but it becomes one when voice commands can be injected at a distance. So this falls into the category of it's unlikely to be a problem even today, really. I mean, it's an interesting attack, but I don't think everyone needs to immediately throw their Google device under a black scarf, although might not be a bad idea.

JASON: Or not in front of a window.

Steve: Right. You make a good point. But of course future devices whose microphones are inadvertently light sensitive might wish to add proactive illumination blocking, since that could presumably be easily done at no cost. And our current crop of voice command devices currently obviously lack that. So it's not something anyone considered. It would be simple to spray a little black paint on the inside of the cavity where that microphone is located so that it could still, you know, still have an acoustic input, but would be blocked by light. That could be done for no cost. It's just not something anyone thought of before. So I imagine, because it won't cost anything to do it in the next rev, I mean, the list of devices, I don't think there was one that wasn't attacked by these guys. I mean, it's every one I'd ever heard of, and some I hadn't heard of. So, you know, just - this is, as I said, oddball, but certainly intriguing.

JASON: Oddball, but not entirely without at least some form of concern. Like you point out, being able to give a voice command that unlocks your door, like that's actually a pretty big deal, I suppose.

Steve: That's not good.

JASON: But then at the same time, who would really go through the trouble? Like I saw the equipment that they used all splayed out onto the floor, and I was just fascinated by what they were able to do with the transducer and everything to kind of convert the signal and everything. It's going to take a really motivated individual. But maybe that's the case for a lot of security flaws. It's not that it's difficult. It's still possible, and therein lies the flaw, so you've got to do something about it.

Steve: Well, and of course, if someone really wants to get into someone's house, you just throw a brick through the window. I mean, it's not...

JASON: Well, there is that, too, yeah.

Steve: It's not like, you know, there's no other way to get into a house other than unlocking the front door. So, you know. And you might say, oh, well, but what about in a really secure facility? Well, then, a really secure facility has no windows. My Level 3 datacenter, where I've got GRC's servers, there's no windows. I mean, it's a bunker.

JASON: Yeah, that's very true, yeah.

Steve: Yeah. So anyway, we have to remember that security is the weakest link, and someone's window is always the weakest link for a break-in. Although the reason it's interesting, the reason we worry about these sorts of attacks over the network is it allows somebody in a foreign country to attack you at distance. This doesn't allow that, either. They can't get you from Russia with a laser beam.

JASON: Yet.

Steve: Yet. So anyway, it's certainly fun to look at the consequences, the actual security consequences of a transducer transducing, or transducting, a form of energy that it wasn't expected to, and how we presume that the transducer's only going to convert what we want. And in this case, not so much.

JASON: Maybe I'll move my Google Home out from the window in my kitchen, just to be safe.

Steve: Well, you don't want it to be in the heat. I wouldn't want it to be in direct sunlight. That's not good for electronics.

JASON: I didn't even think about that. So far so good. It's been there a couple of years now. It's doing okay.

Steve: Okay. So last week's podcast was BlueKeep and DoH. And we've got follow-ups on both of last week's headline topics. Microsoft, their security team believes with good reason, and we'll get to that in a second, that more destructive BlueKeep attacks are on the horizon. And they are once again, for the third time, urging users and enterprises to apply patches if they've been lagging. I think we all know by now that if anyone was going to update any of those more than 700,000 Windows systems which are still right now exposing vulnerable RDP protocol that's susceptible to BlueKeep takeover, it would have happened months ago. Those systems would have been updated. Nobody's going to update those systems, apparently ever.

So I think that these repeated Microsoft warnings must be for political CYA cover, rather than for any practical purpose. You know, they want to be able to say, oh, well, we warned everybody multiple times. We told everybody. We even patched old Windows XP. We did everything we possibly could. I mean, this has to be why they're doing this because they just know, they must know that nobody's listening who's in charge of these 700,000 Windows machines, amazing as that is.

So as we covered last week, we're only seeing BlueKeep leveraged to install cryptominers so far. As ZDNet summed things up in their coverage, they said: "Many security researchers considered the attacks underwhelming and not living up to the hype that was built around BlueKeep for the past six months. This was because Microsoft said BlueKeep" - this is still ZDNet talking. "This was because Microsoft said BlueKeep could be used to build wormable - self-spreading - malware.

"However," writes ZDNet, "the attacks that happened over the weekend" - that's what we talked about last week - "did not deploy malware that could spread on its own. Instead, attackers scanned the Internet for vulnerable systems and attacked each unpatched system, one at a time, deploying a BlueKeep exploit, and then the cryptocurrency miner. This was far," they write, "from the self-spreading malware outbreak that Microsoft said BlueKeep could trigger. Furthermore, in many cases, the BlueKeep exploit failed to work" - well, we'll get to that later because that's been fixed - "crashing systems. But Microsoft says this is just the beginning, and that attackers will eventually refine their attacks, and that the worst is yet to come."

So anyway, Microsoft said last Friday: "While there have been no other verified attacks involving ransomware or other types of malware as of this writing, the BlueKeep exploit will likely be used to deliver payloads far more impactful and damaging than coin miners. We cannot discount enhancements that will likely result in more effective attacks." So, okay. Microsoft tells us again that we all need to update. All of us who can and heard this certainly did already.

So here's an interesting tidbit. We've been hearing a lot about how the attempted exploitation of BlueKeep has been crashing systems. And recall when we talked about this last week that it was when Kevin Beaumont first noticed the BlueKeep exploitation, well, he first noticed the BlueKeep exploitation because the servers of his honeypot farm were crashing. Now we know why. At this time the only public proof-of-concept exploit for the BlueKeep RDP vulnerability is a module for the Metasploit penetration testing framework. That module was in turn assembled from a proof-of-concept code which was donated earlier this summer by RiskSense's security researcher Sean Dillon, Sean who tweets as @zerosum0x0.

So as we know, although the exploit code works, it has had the tendency to crash the systems it's targeting, rather than delivering a remote shell to the attacker, which is what it does when it doesn't crash. And as I just mentioned, it was when 10 of Kevin's 11 RDP honeypots crashed that the world first became alert to this BlueKeep-based campaign. Of course, we often quote Bruce Schneier's astute observation that attacks never get worse, they only ever get better. And eventually, recent events with BlueKeep have followed that path.

This past weekend the BlueKeep Metasploit module was fixed to entirely eliminate the source of the BlueKeep Blue Screen of Death (BSOD) crashes. You're not going to believe what it was that was originally tripping up BlueKeep. It was Microsoft's patch for the Meltdown flaws in Intel's processors. Those systems that were never patched for the Meltdown flaws did not crash. Those that were patched, crashed. And certainly lots of those have been patched, 10 out of Beaumont's 11. Quoting from an interview that Sean Dillon gave over the weekend, he said: "From looking at screenshots of the analysis Marcus Hutchins did" - of course we know Marcus as MalwareTech - "we know code execution was achieved, but that the honeypots were crashing because the exploit did not support kernels with the Meltdown patch installed," said Sean.

The original exploit that he created incorporated what he called a "KVA Shadow mitigation bypass." Those with good memory may recall that KVA Shadow is the official name, Microsoft's official name for the patch for Meltdown. But another researcher pointed out that there was a way to bypass the need for the system call hook that was causing the trouble in the first place. So Sean updated the exploit, which has now been posted to GitHub and incorporated into the updated Metasploit framework which now changes the way this attack occurs so that a Meltdown bypass is no longer needed. This raises the early BlueKeep patch to now a fully weaponized state. It is 100% reliable and no longer triggers BSODs.

So, and I have a link in the show notes for anyone who is interested. There's also a lot of discussion on the various developer forums about this. So essentially in the course of one week, after we saw that this was being used and crashing these kernels, that's been fixed. We now have a 100% reliable non-BSOD-triggering BlueKeep exploit. So if systems were crashing as a result of the implementation of the first exploitation, they will no longer.

And Marcus Hutchins, again, MalwareTech, the guy who, as we know, famously stopped the WannaCry Internet worm in its tracks by registering that bizarre domain name when he noticed from reverse engineering of the worm that it was pinging that DNS, making a query for the DNS, he's weighed in on the issue of a BlueKeep-based worm. He posted the following tweet thread last Friday, which I'm going to share in its entirety since he raises a number of very good points and because he obviously knows what he's talking about.

So his first tweet said: "When the news broke about BlueKeep exploitation in the wild, most of the reactions were basically, 'It's not a worm, so it doesn't matter.'" He said: "I

decided I'd do a thread on why that's wrong, and why a worm isn't even a worst-case scenario."

So he then generated a thread of tweets which read: "There are two main purposes of a worm." And he has in parens, "(self propagation). First, dealing with cases when there are too many vulnerable systems to infect reliably with just scanning alone; and, two, dealing with a large disparity between the number of external- and internal-facing vulnerable systems." He says: "The WannaCry worm served both these purposes. Firstly, there were just too many vulnerable systems to infect by scanning servers." In other words, yes, it was a target-rich environment. It was too rich.

"Secondly," he says, "if a network had SMB" - you know, Server Message Blocks, Windows file sharing - "exposed, then the chances that every single device on that internal network was vulnerable were very high." In other words, if the worm got a foothold on an exposed border server, it could pivot to the LAN and scan internally in order to find probably all the internal systems, which was really what made this thing so destructive is WannaCry penetrated networks.

Then Marcus says: "BlueKeep is different. Not only is the number of externally facing vulnerable machines low enough to infect with a couple of servers, but also RDP is only enabled by default on Windows Server operating systems. Because Windows clients don't expose RDP by default, unlike SMB, a BlueKeep worm wouldn't be able to pivot to systems within a network like WannaCry did. Furthermore, I'd guess it's fairly likely that, if one of the network's RDP servers is exposed to the Internet, they all are.

"For all these reasons, a BlueKeep worm would not be hugely effective and not at all like WannaCry. They might infect marginally, not exponentially, more systems; but the downsides are huge. A worm would not only attract a lot of attention, but be technically challenging due to the limitations of BlueKeep. The exploit" - and he's writing this on Friday before this got fixed. "The exploit is both unstable and non-generic." He says, parens: "(The attacker would need to somehow fingerprint the OS and exploit accordingly)."

He says: "Building a worm in a way that doesn't just repeatedly crash every BlueKeep vulnerable system would be challenging." And of course we know that's been fixed now. He says: "And by no means worth the reward." He says: "I'm not really worried about a worm. What I'm worried about is something that could already be happening. Most BlueKeep vulnerable devices are servers. Generally speaking, Windows servers have the ability to control devices on the network. Either they're domain admin, have network management tools installed, or share the same local admin credentials with the rest of the network. By compromising a network server, it's almost always extremely easy to use automated tooling to pivot internally." He says: "Example, have the server drop ransomware to every system on the network."

He says: "The real risk with BlueKeep is not a worm. A worm is pointless and noisy. Once an attacker is on the network, they can do far more damage with standard automated tools than they could ever do with BlueKeep alone. Remember all those news stories about entire networks being ransomwared. That starts with a single system being hacked. Not even a server. A normal, non-admin, client system. Attackers don't needs worms. It was just convenient in the case of WannaCry/EternalBlue. People need to stop worrying about worms and start worrying about basic network security. Firewall your servers off from the Internet." And maybe he meant Intranet. He must have meant Intranet. He said Internet. Because of course a server on the Internet that's firewalled off from the Internet can't be attacked externally. He says: "Learn about credential hygiene. Occasionally worms happen, but every day there are entire networks compromised using only standard tools."

JASON: That's a good point.

Steve: So I obviously think he's exactly right. We know that since the beginning I had said I didn't think BlueKeep made sense to use with a worm because Shodan already knows, it already has a list of all of the available RDP servers, which have already been scanned many times now to find the ones that are exposing RDP, that has not yet been patched and is vulnerable. So the bad guys already have a list. And, you know, once upon a time you would launch a worm like the original Morris worm when you wanted to hide your identity from the Internet. But now people take over other people's machines in order to perform proxy-based scanning. So you don't even need to hide anymore. So, yeah. I think he's right.

JASON: I would agree.

Steve: So a bunch of great points.

JASON: Absolutely. Smart guy.

Steve: And Jason, we're going to talk about Mozilla and DoH versus Comcast.

JASON: All right. Circling back to DoH. Got a few pieces of news here. Lots of, like, themes in today's show.

Steve: Yeah, we're a theme show. I did just want to - while you were delivering that, I was thinking that I just wanted to make sure that people understood what this means, that we now have a very reliable, non-crashing BlueKeep exploit which will allow a reverse shell to be obtained by anyone on the Internet who wants it for 700,000-plus Windows servers. I mean, that's huge.

JASON: Wow, yeah.

Steve: That's just, you know. And as Marcus says, we don't need a worm. Those machines are sitting there waiting to be commandeered and occupied by bad guys who then sit there and look around and see what they're connected to and what sort of goodies are on the LAN on the inside. So this is, I mean, this is bad. And so Microsoft is warning people, as I said, I think because they have to. But it's been months now, and these systems are not getting themselves patched. They may be running cryptominers today. They're going to have bad guys setting up shop and looking around to see what other mischief they can get up to shortly. It's just a matter of time. So, wow.

You know, and I just realized there's another name for Comcast. I should have - we should be calling them "Concast."

JASON: Concast. You can't be the first to come up with that one. But it makes sense.

Steve: Mozilla versus Concast. I was really glad to see Mozilla take the leaked slide deck which we discussed last week and push back hard with their own clarification and much more accurate letter to Congress. Their letter was signed by Marshall Erwin, who's the Senior Director of Trust and Security for Mozilla. I've got a link to it, if anyone is interested. And I'm not going to go through the whole thing. But I will share enough of the beginning of the letter so that our listeners get a sense for Mozilla's take and their position and because they really raised some very good points.

He wrote: "Dear Chairs and Ranking Members." And this was an open letter addressed to Congress. "We are writing to express our concern about the privacy and security practices of Internet service providers (ISPs), particularly as they relate to the domain

name services provided to American consumers. Our recent experience in rolling out DNS-over-HTTPS (DoH), an important privacy and security protection for consumers, has raised questions about how ISPs collect and use sensitive user data in their gatekeeper role over Internet usage.

"With this in mind, a congressional examination of ISP practices may uncover valuable insights, educate the public, and help guide continuing efforts to draft consumer privacy legislation. During the last two years, Mozilla, in partnership with other industry stakeholders, has worked to develop, standardize, and deploy DoH, a critical security improvement to the underlying architecture of the Internet. A complementary effort to our work to fight ubiquitous web tracking, DoH will make it harder to spy on or tamper with users' browsing activity and will protect users from DNS providers - including ISPs - that can monetize personal data. We believe that such proactive measures have become necessary to protect users in light of the extensive record of ISP abuse of personal data, including the following incidents." And then they highlight four goodies.

"One, providers sold the real-time location data of their mobile broadband customers to third parties without user knowledge or meaningful consent. In one particular case, an intermediary was found to be selling particularly sensitive GPS data, which can pinpoint the location of users within a building for over five years. Two, ISPs have repeatedly manipulated DNS to serve advertisements to consumers. Comcast has previously injected ads to users connected to its public WiFi hotspots, potentially creating new security vulnerabilities in websites. And last year, CenturyLink injected ads for its paid filtering software and disabled the Internet access of its users until they acknowledged the offer.

"Verizon tracked the Internet activity of over 100 million users without their consent through supercookies that could not be deleted or circumvented. This allowed Verizon to closely monitor the sites that users visited and catalogue their interests without their knowledge. And, finally, AT&T operated a program that required users to pay an extra $29 per month to opt out of the collection and monetization of their browsing history for targeted ads. While the company ended the program after public criticism, it has considered reviving it in the current deregulated environment."

So they said: "Unsurprisingly, our work on DoH has prompted a campaign to forestall these privacy and security protections, as demonstrated by the recent letter to Congress from major telecommunications associations. That letter contained a number of factual inaccuracies. These have been examined in detail by others and as such will not be given an in-depth treatment here. Nonetheless, it is important to highlight the underlying premise of that letter. Telecommunications associations are explicitly arguing that ISPs need to be in a position to collect and monetize users' data. This is inconsistent with arguments made just two years ago regarding whether privacy rules were needed to govern ISP data use.

"With the 2017 repeal of the Broadband Privacy Order, a substantial gap in consumer privacy protection was created. That gap exists today. ISPs are people's gateway to the Internet. That gateway can serve as a data collection point, providing ISPs with unique access to sensitive browsing information. That is why broadband privacy rules would have required ISPs to get the clear consent to use and share their subscribers' information. However, those rules are no longer in place."

And I'll conclude my quote of this, although their letter went on: "Our approach with DoH attempts to close part of this regulatory gap through technology and strong protections for user privacy. Mozilla's policy establishes strict requirements for potential Firefox DNS resolvers, including requiring that data only be retained for as long as it is necessary to operate the resolver service, that data only be used for the purpose of operating that service, and that partners maintain a privacy notice specifically for the resolver that

publicly attests to data collection and policies. Unfortunately, ISPs do not maintain privacy notices for their DNS services. As a result, their policies are opaque to users. It is unclear what data is being retained, how it is being used, or who it is being shared with."

So bravo to Mozilla for pushing back on the nonsense that we read in what Concast shared with Congress and in this campaign that they are pushing for this kind of, well, just basically saying we don't want this to happen, and we're not happy, is what it boils down to.

JASON: Now, lucky for Comcast, Concast is actually a business. It exists. So they're like, ha ha, you can't even get us with that one, Steve.

Steve: Although who would want that name? Who'd want to call themselves Concast? I don't know.

JASON: I know.

Steve: Not one I would go for.

JASON: Yeah. It's like - I'm trying to figure this out. It's like construction solutions. It's completely unrelated. Anyways, I was very curious. I was like, this has got to be out there somehow.

Steve: So what I found, somewhat to my surprise, is that despite ISP associations' hissy fits, DoH will be rolling out for all major browsers. In the wake of all this DoH noise, ZDNet took it upon themselves to interview the product managers of all six major browsers. In alphabetical order they are Brave, Chrome, Edge, Firefox, Opera, and Vivaldi. And this was where I was a little surprised. Although I guess in retrospect I shouldn't have been because so many of them are now Chromium based. The good news is every single one of these top six browsers already has or soon will have DoH support. All of them.

Brave, the Brave browser. Tom Lowenthal, Product Manager at Brave for Privacy & Security, told ZDNet: "We absolutely want to implement it. Implementing DoH is far more than just the technical work, though. We need to decide on sensible and protective defaults for the vast majority of people who don't think about their DNS configuration while making sure that we don't break things for the people and organizations who have carefully tuned their setup." Because Brave is built on top of Chromium, which of course as we know, well, Chromium's open-source browser codebase, DoH support is available today. However, the Brave team has not yet tweaked the feature so that it works exactly the way they want to, so it's not enabled by default. But it's there. You can go to brave://flags/#dns-over-https and enable it right now, if you're a Brave user, if you choose to. So bravo for Brave.

Chrome, of course, as we know, does have it available. It's not yet enabled for everyone by default since Google is currently running a limited experiment with a small number of users to see how DoH fares in a real-world test. And as we've noted, they're taking an adaptive approach, first attempting to honor the user's existing DNS provider to see whether it supports DoH, and using it, if possible. If not, then it follows a number of sort of heuristic rules of thumb to figure out what it should do. And again, in Chrome, you say chrome://flags/#dns-over-https if you want to take control of it yourself and turn on DoH right now.

Similarly with Edge. Even Microsoft told ZDNet that they were supportive of DoH, but they couldn't share their exact plans. So, yeah, it sounds more like Microsoft. However, like Brave, and as we know, the soon-to-go-mainstream, in mid-January, Chromium-based version of Edge already supports DoH. And the preview versions do. Same flag:

edge:// and then the rest. And you can turn DoH on in Edge right now. There are some additional tricks, command line switches and things, and I have links to a blog posting by one of the Edge developers, if anyone is a committed Edge user and is interested in getting going with DNS-over-HTTPS in Edge right now.

And of course, as we know, Firefox was the first out of the gate with DoH and as a consequence took some undeserved, in my opinion, arrows in their back for simply standardizing upon Cloudflare as their DoH provider. No one really took the time to understand how rigorously Mozilla had vetted Cloudflare. And many people who don't listen to this podcast might mistakenly believe that Cloudflare is just some other CDN. But anyone who has the class to erect a large wall of lava lamps and use the video image of those lava lamps to generate true random numbers definitely stands out as an innovator in my book. And of course we know that Cloudflare did that. And they have met all of Mozilla's requirements that we noted above in their letter to Congress of really respecting user privacy and absolutely never using it for any sort of monetization scheme. And since Firefox is officially supporting it, you can turn it on in the standard browser UI.

Opera has already rolled out DoH support. It is disabled by default for all users, but it can be enabled at any time in the stable release. And it works without going through any additional steps. They achieved the "no additional steps" in the same way that Firefox did, just by standardizing on Cloudflare's 1.1.1.1 DoH resolver. And I'll note, though, that as Opera users probably know, Opera builds in a popular VPN. It and DoH, the VPN in DoH are not compatible with each other. On the other hand, if you're using a VPN, you already have a privacy encrypting tunnel to route all of your browser's queries. So assuming that DNS goes through the VPN tunnel, I hadn't thought of that until just now. That would be worth checking to see whether Opera's DNS with the VPN up is going through the tunnel or not. I don't know either way. In Opera, it's opera://flags/opera-doh in order to get to its configuration.

Apple, with Safari, was completely mute. ZDNet was unable to obtain any reply from Apple about Safari one way or the other. But they are certainly privacy focused and user feature privacy focused. So I think once the dust settles on DoH, Safari will probably follow suit in what will end up becoming probably the industry norm for browser-based DNS queries.

And lastly, Vivaldi. It, too, is Chromium based. So it also works like Chrome. And its flags are like the other Chromium-based browsers: vivaldi://flags/#dns-over-https. So among many of Comcast's misassertions was their focus upon Google. I guess Google makes a convenient soft target lately. But as we know, and as this list just demonstrates, despite Comcast's assertions, it's not just Google, but the entire browser industry that is heading to DoH just as fast as development cycles and field trials will allow.

So I think it is very clear. ISPs ought to be reading the handwriting on the wall, which is that browsers are going to be privacy protecting their DNS because they can. And so I think what will happen is we'll have ISPs, I'm sure right now they're scurrying to bring up their own DoH support on their own DNS servers so that then they, too, will be able to be the DoH provider for browsers, which will return them to having an ability to see into their customers' DNS queries. At which point hopefully browsers will make it easy to say, you know, I want to override my ISP's DNS. I want to go to a trusted provider that is not going to be monetizing me without my knowledge or permission. So I'll bet you that we see that before long. And again, there are many providers of DoH beyond Cloudflare. Mozilla just decided to settle on them because they were able to get the commitment from someone they trusted. And I certainly trust Cloudflare, as well. And I think our listeners could, too.

JASON: Absolutely. Absolutely. And is there much that the ISPs could really do? I mean, at this point, if all the browsers are implementing it, Comcast is just relying on their hopes that the government might somehow step in, in some way, shape, or form, and prevent this.

Steve: It's true. There's nothing that they can do. I mean, they could block it. But that would be the end of life. That would be, yeah. That's not going to happen. So they can't see into it. I mean, it's worth noting that DNS was a simple way to see what people were looking up. But people are still using the IP addresses they get from DNS to make the connections. So it's more work for an ISP to use the IP addresses and reverse lookup those IP addresses to see who they're connecting to. And it's not also, you know, many hosted sites will share many domains across a single IP. So you can get some confusion. There is some ambiguity if you use IPs. But it's still something that ISPs will always have access to, unless their customers use a VPN, which then shields that, too. But that seems unlikely.

So, yeah. ISPs really can't do anything. I think what we're going to see them do is, since Chrome is the majority browser, since it defers to using the preconfigured DNS resolver by default, ISPs are going to bring up their own DOH resolvers in order to once again capture this information, and sort of rendering this whole exercise moot. But users who can override and choose to override their built-in DNS - and maybe Firefox will continue to say, hey, we're using Cloudflare. We want to protect you from ISPs even after they bring up their own DoH services. Which Firefox's strategy does; Google's strategy doesn't.

JASON: Right.

Steve: So that'll be interesting.

JASON: Yes.

Steve: And this little tidbit, this was just a throwaway.

JASON: I know. You're speaking to me with this one, by the way.

Steve: Oh, no kidding. Interesting. I noted to our listeners when I saw this appear. We noted on this podcast when Microsoft decided to add the user's Downloads directory to their handy little Disk Cleanup applet in Windows 10. And I specifically cautioned our listeners to be careful with that one, since on the one hand many people are in the habit of forgetting their Downloads folder and just allow it to grow without end. I happened to note the other day that Lorrie's is like, I don't know how many gigs of stuff. Everything she has ever downloaded is in that folder. Which, I mean, I'm not kidding you. And I keep thinking - I keep looking at it and thinking, oh, this hurts. But I'm not going to mess with her machine. And as a consequence it grows without end.

And many of today's downloads are massive things that just sit there and squat on space. So I could see, as a consequence of that, I could see the logic of having Downloads in the Disk Cleanup applet. And in this era of multi-terabyte storage, many people have taken to using their Downloads directory, kind of more or less deliberately, as an archive, a massive archive of everything they have ever downloaded. And they assume that everything will always be there. So Microsoft apparently discovered that too many people were selecting, simply going through and selecting all of the Disk Cleanup category checkboxes without looking and were, as a consequence, permanently and inadvertently deleting their download archives. So that feature, which was first added into last year's ill-fated - and it then became infamous - October 2018 Windows 10 update, is now being removed.

So in the future, if you want to delete stuff from your Downloads directory, you'll need to do it yourself. And really, I think that does make the most sense. Disk Cleanup should be for those obscure OS-level things like in the list: DirectX Shader Cache. Okay. Delivery optimization files, whatever that is. System error memory dump files. Good, get rid of it. And Windows Update Cleanup. I looked as I was putting my notes together last night. I've got a gig of that on the machine that I was using last night. So good riddance. After I'm sure that I've updated and everything is stable and so forth, get rid of it. That crap accumulates that no one needs. Sort of like OS belly button lint. So I can understand that being removed. And so what happened with you, Jason? You got bit?

JASON: Well, no, no. It was just as I was reading through this I was thinking about how I use my Downloads folder, or rather how I don't...

Steve: It is convenient to go back and get something sometimes.

JASON: I mean, yeah, it is. I don't know if it's the most secure approach. But, yeah, I kind of compile things in my Downloads folder. I don't think of anything that's in my Downloads folder as permanent. But I do keep things there indefinitely. And, you know, it might be years down the line where I, like, do a search for something, and what do you know, it's already in my downloads folder instead of me having to go out and download it again. So it comes in handy.

Steve: It's very much like when you go to Amazon to buy something, and it says, "You purchased this on such and such." And I go, oh.

JASON: Oh, did I?

Steve: I've got that around here. I've got that around here somewhere.

JASON: Or you download something, and then you realize the file that's been saved is a (2) or a (3).

Steve: Yes.

JASON: And you're like, oh. I guess I already did this.

Steve: Yup.

JASON: But, yeah. So I'm just - I never thought about how I use my Downloads folder, whether it's a good or a bad idea. But, yeah, it's like a growing collection. Every once in a while I'll jump in there, I'll be like - I'll get, like, creative or get a cleaning kind of a bone in my body and clean it up a little bit. But I end up keeping things. But then I don't think about it as permanent storage anyway. So I don't know. If software on my computer suddenly just assumed that my Downloads folder was a place for Disk Cleanup to automatically wipe away, I think I would be a little bit upset if I didn't realize that, and then suddenly it was empty. I'd be wondering what I was missing.

Steve: And I'm going to stop making fun of Lorrie. I just looked at mine: 3.576GB.

JASON: Oh, really. Interesting.

Steve: Okay. Oh, who's talking, 3.576. Yeah, thank goodness I don't have that checked on my machine. Whew.

JASON: But the question is, could you, without looking at the contents of your Downloads folder, empty it right now? Would you be uncomfortable with that?

Steve: Uncomfortable, yes.

JASON: Uncomfortable. So would I.

Steve: There's definitely - I'm sure there's things. Now, what I told her I thought we should do for her, and now I'm thinking, okay, Gibson, take your own advice, is just grab the whole thing and drag it over to the Drobo so it's there.

JASON: Right.

Steve: And it's not on, I mean, like right now I'm making images of all that crap every single time I do an image of my whole system, which I have automated. And that's dumb.

JASON: Yeah. Yeah, it's true, moving it over to some sort of archive or whatever. I tend to collect files on my desktop, as well, at home.

Steve: Oh, I can't find my desktop. I can't even see it. It's just covered with little squares.

JASON: I don't go that far. But at some point it gets messy enough that I'm like, all right, I don't have the time to pull this stuff out now. But I do want to save it. So it's like, goes into a folder called "My Desktop" and the date. And then that goes into some archive somewhere. And then I keep it forever and never turn to it again.

Steve: Yes. Jason, on the wall behind me there are books. I actually - I have books.

JASON: What are those?

Steve: You know, like "Programming the VGA." And it actually turns out...

JASON: When was the last time you pulled out that book?

Steve: Well, it actually was for SpinRite 6 I needed to do some VGA - because I had to program it myself. But that's my point.

JASON: Still, they look really nice.

Steve: Well, here's - I'm looking at a book on CGI. Yeah. And Microsoft AFC. Yeah, I could really do with some housecleaning.

JASON: That's okay. No, keep it. Keep it. It helps your studio stay soundproof.

Steve: That's true. That's true.

JASON: That does help. It does help.

Steve: It's not a hard surface so much.

JASON: That's right.

Steve: So one last little tidbit for our Linux and BSD users. The compression library included by default in Debian, Ubuntu, Gentoo, Arch Linux, FreeBSD, and NetBSD distros - and others, probably all of them - contains a vulnerability that can allow hackers to execute code on user machines. It's not an across-the-network vulnerability, so it's not like a BlueKeep thing. But last week, even though this thing's been known since June,

last week details about this major bug impacting essentially Libarchive, which is the archiving library that all of these OSes use, went public. But it was held embargoed until all the Linux and FreeBSD distros had rolled out updates containing patches for the version of Libarchive they had been shipping.

So of course we know a decompressor is an interpreter because it interprets the tokenized compressed representation of a file and reexpands it based on the tokens that it receives in the byte stream of the compressed contents. It is extremely difficult to find every possible flaw, and decompressors especially tend to be even more difficult than other interpreters because they must inherently operate with fewer constraints. They just sort of have to believe the data that they're being given.

So this bug, which is being tracked as CVE-2019-18408, not surprisingly allows an attacker to execute code on a user's machine if the user's machine attempts to decompress a deliberately malformed archive file. So exploit scenarios would involve the reception and decompression, you know, like viewing something from email, receiving something in email, maybe downloading something from the 'Net one way or another, from a web or whatever. It's not something that allows a remote attacker to get you. But if you run something which has been deliberately created to exploit this flaw on a Linux distro or FreeBSD or any BSD that has not been updated, you could get owned. So as I mentioned, it was originally discovered and patched back in June. The vulnerability was identified by Google security researchers using some automated code testing tools, one they call, and I'm not kidding, ClusterFuzz and OSS-Fuzz. It was patched in Libarchive 3.4.0. I went over to Libarchive.org and noted that 3.4.0 was released in June, on the 13th of June. So it has indeed been out there for a long time.

Our takeaway for everyone is that now would be a good time, if you haven't for a while, to resynchronize your OSes with the latest repository updates. You may already have version 3.4.0. You might just do a - presumably you could do a man Libarchive and see what version of Libarchive you have on your system. As long as you're at 3.4.0, you're okay. If not, you want to update because your system is vulnerable should you receive something. And it's not going to be a widespread attack. It's not going to be a spray. There are no known exploits in the wild, though they would likely be targeted. And so you don't want to be a victim of that kind of targeting. So just make sure you have v3.4.0, which as I said has been around for a year, I mean since June 13th. So you may well already have it.

JASON: And now a deep dive on Credential Delegation. Tell us what this is all about.

Steve: So as we know, web servers prove the domains they control by providing a certificate which has been previously signed by a certificate authority whom the client knows and trusts. That certificate will contain an enumeration of one or more Internet domains and/or IP addresses and the server's previously assigned public key. During the TLS handshake, the client challenges the server with a random nonce, which the server must sign using its matching secret private key. The client uses the public key in the signed certificate authority certificate to verify the server signature of its message containing the nonce. And if the signature verifies, the client can then be sure of one thing, that the server is indeed currently in possession of the private key that matches the public key in the certificate that was signed by the trusted certificate authority.

So that's how our certificate system works today. But what if the server's private key is ever stolen? That's, as we've often talked about it, that's the Achilles heel of this system. And it's such a significant and intractable problem for our industry that I and this podcast have spent a great deal of time through the years exploring the problem and sharing the problem and its various solutions with our listeners.

And in terms of the scope of the problem, it's one thing for me to keep GRC's private key secret. I have one 1U high rack-mounted server that's as well shielded from external network penetration as I've been able to make it. That physical server is in a locked rack to which only I and Level 3 have access. As I mentioned, Level 3's facility is a bunker. Neither Sue nor Greg have ever had any reason to have access, so I'm the only one who knows it, who has access to it. That server is in a locked, heavily monitored and guarded, super-secure facility, and there are no other copies of that private key anywhere in the world. So it's about as secure as I can imagine.

But consider the situation for Facebook or Cloudflare or Google or any other massive Internet provider. They have thousands of servers spread across the globe in hundreds of datacenters. And they have hundreds of thousands of employees. And sure, many fewer than all their employees need to have access or do have access to those servers. But certainly still a great many. We recently heard that Twitter had discovered a couple of spies in their midst. So that's a problem. And so, too, is keeping an absolute secret, which is what that private key must be kept, an absolute secret across thousands of networked servers around the world. You know, mistakes happen. We cover them often on this podcast. This is, you know, it could be called "Security Mistakes" as well as "Security Now!."

And so we know that robust security is inherently always a best effort. We make the best effort we can to eliminate all the bugs in our software. When they are revealed, we fix them, hopefully, as quickly as we can to minimize the damage. So the point is that resiliency in the face of a breach, which is to say recovering gracefully and quickly, is often the best we can do. So when a server's private key escapes through whatever means and for whatever reason, the best thing that can be done is to immediately revoke its authority to represent the property for which it was signed so that it cannot be used to impersonate any of those properties moving forward.

All of our long-time listeners to this podcast will already know about revocation, this process of revoking a certificate's authority. Revocation's utter failure and total implementation collapse has been one of my personal crusades and hobbyhorses for years. Everyone will recall that I invested a great deal of time and trouble to demonstrate just how utterly broken - nonexistent, really - Google's Chrome browser's CRLSET revocation handling has always been. I went so far as to deliberately create a revoked certificate and set it up on a revoked site to demonstrate that Chrome would honor it on every platform, despite the fact that it was clearly revoked and should never be honored. What did Google do? They added a special case exception for that particular revoked certificate. So I changed the certificate, and they gave up because they realized I would just change the certificate again. The point is Chrome has always been broken. They don't have revocation.

Now, since all of these certificates, all of the certificates that are signed, as we know, have an expiration of two to three years for public domain TLS certificates. They will eventually take themselves out of service by self-expiring. So it's only the still valid by date certificates which we wish were no longer honored that we need to somehow suppress. So the first solution, the first of many bad solutions, was CRLs, Certificate Revocation Lists. Certificate authorities were supposed to publish a list of all the certificates they had issued which had been revoked, but were otherwise still valid. And then anyone relying upon a certificate authority's certificate was supposed to check that list before trusting the certificate.

But as it turned out in practice, this didn't work very well. Certificate authorities were slow to update their lists, when they even bothered to at all. Those lists would become quite long and thus took a while to download. So browsers didn't want to be doing that all the time. And so the relying clients that were relying upon the certificates weren't much better about pulling and checking the lists. So even when things were being

checked, which wasn't really that often, so much so that it's just kind of broken completely, even in the best case there would be a significant gap from the time somebody, a CA would revoke a certificate, and it would actually be known to be revoked, if ever.

So the next thing that we came up with was OCSP, stands for Online Certificate Status Protocol. That was a nice solution that was intended to close this large window of time between a certificate's revocation and a client's awareness of that having happened. But that system, too, has flaws that we've discussed in the past. Checking the status of every TLS certificate takes additional time that no one wants to spend. So you go to a site. You obtain a certificate from the browser. Now the browser has to look at the certificate, find in the certificate's metadata the URL of that certificate, the issuing certificate authority's OCSP server, make a connection to the OCSP server, querying it with the OCSP protocol, for the updated status of the certificate that it's just received.

The reason you do this is that it might be a bad guy who has sent you the certificate. It has since been revoked, but the certificate itself hasn't expired because it could last three years. And so this gives you a real-time means of getting the current revoked status of the certificate. But that's an extra connection. Some CAs' OCSP servers, certainly in the beginning, were slow. Sometimes they never replied. They were offline. And so then the browser doesn't get an affirmative okay. What does it do? Well, it's going to either fail open or closed. That is, it's either going to fail and trust, if it can't get an affirmative denial, or it's going to fail closed and refuse a connection. But the certificate might be valid. It just can't verify that it's valid.

So if the browser chooses to fail in the trusting direction, that means that a bad guy can block the OCSP query. I mean, presumably if a bad guy is in a position to spoof a remote server anyway, that sort of suggests they're in the position to intercept the user's traffic. And so they could block the query to the OCSP server, knowing that the browser will fail in the trusting fashion, and then the bad guy wins. So OCSP has its problems.

For all these reasons, the one correct and perfect solution is OCSP Stapling. And when we last looked, Nginx had some flaky behavior with OCSP Stapling. I love this solution. Now what happens is the server which is sending the certificate staples a recent OCSP assertion to their certificate and provides it to the client. The client looks at the certificate, says oh, look, I have the OCSP assertion, which is fresh, that is, maybe as fresh as a few hours old, which has been signed by the certificate authority.

So what that does is it shifts the burden of providing a fresh assertion onto the site that you're already connecting to, getting the certificate from. And that makes a lot of sense. The server periodically goes out and updates its own OCSP assertion, which it sends out with all the certificates. And, for example, GRC does that. So my sense is, moving forward, once this is well supported by all the servers across the 'Net, it really represents a good solution. But we're still not quite there, and there are some use cases that this doesn't fit perfectly. And, well, for example, the one that we're about to address here.

Lastly, though, there's something that's happened since. If the web server is going to be taking the trouble to periodically obtain a fresh assertion of its certificate's validity from its certificate authority, why not just allow the server to obtain a completely new and fresh certificate with a short lifespan? And that of course is one of the features of the industry's latest innovation with the so-called ACME protocol, A-C-M-E, which stands for Automated Certificate Management Environment. As we know, ACME first appeared with the Let's Encrypt movement, and they were followed by other CAs, including now my favorite one, DigiCert, that also offers ACME-based certificates. ACME gives us a sort of workable compromise.

But there's a frightening tradeoff between safety and failure. Let's Encrypt's certificates are valid for 90 days. So that's a lot better. Which is to say they have an expiration date which is when the certificate is issued, no further than 90 days in the future, the idea being that, since it's an automated system, it's easy to update the certificate before then. But still, that's three months. So it's a lot better than three years of validity, but it's still a large open window. If a recently issued Let's Encrypt certificate were to be compromised, that compromised certificate will have an expiration date nearly 90 days away. So during that period of a time a great deal of damage can be done.

If we were to automate the issuance of very short-duration certs, like hours instead of months, then the good side is, the upside, is that, if one of those were to be leaked, it's a very low value. It's going to die almost before the bad guy is able to get set up and ready and try to use it. But then, with very short duration certificates, we face the possibility of the dynamically issuing certificate authority suffering any sort of problem that could interfere with our refresh. They might be DDoSed or suffer a network outage of some sort, during which all of everybody's short-duration certs would expire and could not be replaced, which would lead to a catastrophic loss of service. So that's no good, either.

The solution to this problem is a new, in-the-works system formally known as Delegated Credentials for TLS. It's been quietly in the works for the past three years by Facebook and Cloudflare, who have this problem, and Mozilla, who is over on the client side. Cisco also had some involvement. And this has just surfaced: Delegated Credentials for TLS.

Now, because it requires support from our browsing clients, if you think about it, that's the one thing that Let's Encrypt doesn't require. Let's Encrypt needed to be trusted in the beginning. So as we know, they were using cross-signed certificates. So I think it was Global Trust or somebody was signing, or Global Sign, somebody was signing their certs in the beginning until the Let's Encrypt root was readily available in all browser clients, as it is now.

The point is that none of those earlier solutions required client-side support. This does. This is an evolution of TLS v1.3 to add an extension to the protocol which does need awareness in the client. And as I mentioned at the top of the show, Firefox has it now. And our listeners can play with this immediately. So because it requires web browsing client support, it's going to take some time to become fully deployed. But it does offer a new and very useful tool that would be practical when dealing with these high-risk scenarios involving many thousands of servers that all need to keep a secret.

It wouldn't be of any use to me, for example, where I've got one server with one cert that is locked away as securely as I've been able to make it. But in the case of Facebook or Cloudflare, where they've got to have certificates spread around in order to quickly terminate TLS connections, yet they're at risk of that facility being compromised or a server being compromised in any way, they need a way of offering short-duration attestations, or in this case delegations, of their longer lived credentials. And that's what this system provides.

We already have the well-established concept of a certificate hierarchy, where the CA signs typically now an intermediate CA certificate, just so that their grand master can be locked away in deep freeze. Typically an intermediate certificate is there. It then signs the certificate which is being presented to clients by the web server. And the idea is that this is a hierarchy because each certificate is signed by the one above it.

What this new delegated credential's extension to TLS 1.3 offers for the first time ever is the ability for the end certificate to delegate its credentials to a short-duration certificate. That is, essentially the likes of Facebook and Cloudflare would be able to sign this credential delegation themselves and include in that the public key for a short-lived private key. The browser would use the public key in that credential rather than the

public key in the parent credential, which the delegated credential-using server would no longer honor. It wouldn't even have the matching private key. It only is able to have the public key. It would sign that credential and offer it to the server.

So, okay. So let me step back a little bit. The web browser client needs to be updated. Firefox is. When you enable its credential delegation, all of its connections have this delegated credential acknowledgment added to the Client Hello packet, which is the first thing that goes off to the remote web server as it's doing its TLS handshake. That tells the server that the client is up to speed and is aware of the operation of credential delegation. So the server that it's connected to does not have the master credential. It does not have the private key which the ultimate certificate authority issued. Instead, it has a short-lived certificate which it received from a master server at Facebook or Cloudflare.

So it's sort of like a CA within the organization which is able to delegate its credential for a short period of time without ever giving out its private key. That is, that private key never leaves headquarters. Only the short-lived credentials which are signed by it leave. And they contain a similarly short-lived private key. So the credential contains a public key which the browser will use to complete its handshake, an expiration date for this credential, and the signature of the delegated credentials which were signed by the server's certificate which never leaves headquarters.

So this re-uses the proven strength and integrity of the existing certificate hierarchy by allowing content providers themselves to provide the last link in the certificate chain for the issuance of these short-duration TLS certificates. And there are a couple interesting little twists here. First of all, there is no provision for revocation. They're not even trying anymore to revoke. There's no provision for the revocation of these delegated credentials.

On the other hand, they are hard-limited to a maximum validity duration of seven days. And any client must, as part of the protocol, must and will refuse any delegated credential that had a life longer than seven days. So when the client gets it, you know how all certificates have a not-valid-before and not-valid-after dates, a date of first valid and last valid. If that is any wider than seven days, the certificate is deemed invalid. So no one will bother issuing those because no clients will accept them.

And so that means that essentially we've come up with a solution. And who knows. Maybe - it's not like seven days is what these guys will actually choose. It's that they can. That is, the spec says they could be up to seven days. It may very well be, once the system is up and running and is proven, and they've got backups for their master, their grand master, and so forth, they may choose to set them for four hours. I mean, they can be whatever they want them to be up to seven days.

So this is really cool because it solves the problem of the CA being offline, that is, the problem that Let's Encrypt might have of relying upon an external party over whose network you have no control by getting long-lived certificates from them, and then keeping it within the family, keeping it within your own local network, essentially, with a network you do have control of, where a single master server, which keeps the master key for one of these major content providers, is then able to reissue its own very short-lived delegations so that it's never exposing its master private key. It's only exposed essentially a working key good for however long the provider chooses to have it be true for.

So this is already in Firefox. I think it is a forthcoming IETF standard, been well designed. I've got links to the standard in the show notes, and something our users can try. If you are a Firefox user or have access to Firefox, and you're curious, go to about:config. We know that the result is an overwhelming list of tweaks and options and settings. So you

want to use the search bar and just put in "delegated," or "delegate." That's enough. I put in "dele," and it whittled it down to about six different hits. But if you put in "delegated," you get one. You will see there Firefox's support, right now, for this credential delegation. Mine was turned off, so I double-clicked it to flip it to "true" to turn it on. Then I went to "fb," as in Facebook, fbdelegatedcredentials.com. That's a nice little test site which anyone can use: www.fbdelegatedcredentials.com. And if everything works, you'll just see a notice that acknowledges that the server and your browser have just successfully negotiated a TLS v1.3 connection successfully using delegated credentials.

So as I said, this doesn't make sense for single server scenarios where you already only have your key in one place. It's not really buying you something. I mean, I guess I could see you might put the key somewhere else, and have that even more secure, and have your server reach out for an updated delegated credential for it to then resend to all of the clients connecting to it for some period of time. Yeah, maybe.

But, boy, this is a beautiful solution for what we've seen evolve on our globe, which is these massive content providers that are globe spanning, that want to have local points of presence for their service offerings in datacenters not far away from their customers, yet they've got this management problem of being a catastrophe to be issuing, to be using certificates with multiyear expirations against the danger of one of those private keys, any one of them anywhere, getting lose, and the revocation system being so broken that it might as well not even - no one should have bothered with it in the first place. Which actually, comically, is the path that Chrome took. It was like, well, it's broken, so we're not going to bother. It's like, well, okay, but it has been useful. Anyway, a very cool solution. And now all of our listeners understand it.

JASON: So and then, as you spelled out, Firefox already has support. You can flag that on. Chrome going to circle back around on this?

Steve: Oh, no question.

JASON: No question, it's just a matter of when.

Steve: And as a consequence of the fact that it's pretty much Firefox and Chromium, what it means is that as soon as the Chromium code gets it, all the other browsers get it.

JASON: Yeah, that's true.

Steve: Edge, Opera, Brave, you know. And I'm sure that Safari will do it. I mean, and of course you do need a fallback for when a client doesn't support that. Then it just doesn't know about delegated credentials. It'll go, what the heck? So there will be a phasing over time. But here we are, I mean, it's not difficult for it to support it. It's a matter of the client adding that to its Client Hello handshake. You know, Firefox is all open source, too. And the Chromium guys and Mozilla are always sharing stuff.

So, I mean, it's probably already in the works in some early build of Chrome, if it's not already there. I don't know for sure either way. But it's as simple as a little bit of logic to add the "I know about delegation" flag on the Client Hello packet. And then, when you receive the response, you look in a different place. You don't use the public key that is in the certificate that you received. It will not be honored. You use the public key which is in the delegated credential, which is signed by the public key, you know, the master public key. And so you use that public key to verify the validity of the delegated credential.

So again, it's hard to describe in language. I hope I've done so for our listeners. But basically a relatively simple change for our clients. The good news is in the industry we really only have three now: Firefox, Safari, and Chromium, which is an umbrella for all of

the others. And then we're going to have this pretty quickly. And as soon as it's ubiquitously present, then Facebook and Cloudflare and the others will be able to withdraw their private keys from all of their end node servers and instead start having them only serve very short-term credentials. And what it really means is that they're not going to be attacked because they won't have a credential worth attacking.

JASON: Right.

Steve: So it also lowers the pressure on getting one of those prized private keys out of one of those servers. It's a very cool, like next-generation change. So bravo to those guys who put this together.

JASON: Yeah, it's definitely progressive. I love it. And someone in chat, there was a little bit of confusion. It's fbdelegatedcredentials.com, not fb.delegatedcredentials. So all one word.

Steve: Yes. One long...

JASON: And that should take you there and check it out later.

Steve: Yes, good, thank you, fbdelegatedcredentials.com.

JASON: Right on. Interesting stuff. That's something to look forward to. And, yeah, great show, Steve.

Steve: Well, our listeners have been wanting a little something meaty for a while. So I figured, okay. We'll wind them around that one.

JASON: Love it. Also, just a totally unrelated aside, I was noticing as we were reaching the end of that doc, I was looking at your SQRL logo, and then also on your mic. It just kind of, you know, that's a really brilliant logo. I have to say I really like it a lot because the longer I look at it, the more I realize what was probably obvious to so many people, how the tail, yes, looks like a lock, but also is in the shape of an S for SQRL. And I don't know, whoever did your logo design for SQRL is doing a really a great job.

Steve: It was a really gifted, very temperamental Spaniard who - I used the LogoSauce.com site, which I've used them a number of times. They're a great site. But, boy, this guy, he just, I mean, I used the word "hissy fit" earlier. And, you know, he would get all bent out of shape and go away, and he was worried that other people were stealing his ideas because I was trying to move the whole community in the right direction. And anyway, he ended up, I ended up paying him for it, and I'm very happy for it. And of course I put it in the public domain. It's everyone's to use who is doing SQRL things. But anyway, thank you. I'm glad you like it. I am just so pleased with it.

JASON: It was just in that moment that I realized, like I've always seen the lock and the tail, but it was that moment that I realized, wait a minute, it's also an S. That works for Steve. That works for SQRL. This is amazing. And of course speaking of SQRL, we do have the SQRL event coming up November 30th. It's just a couple of weekends away. Saturday, November 30th, 2:15 p.m. Pacific is when that's going to take place. And I'm sure that's going to be live streamed, of course, TWiT.tv/live, if you can't be here in-studio. If you can be here in-studio, I'm super not aware of whether there are even seats left at this point. But tickets@twit.tv.

If you know for certain that you can make it, email tickets@twit.tv so you're not taking that opportunity from someone else who could possibly do it. And let them know. Inquire within, and they'll let you know if there's still space. I have not heard this past week

whether that has filled up entirely or not. But that's Saturday, November 30th, 2:15 p.m. Pacific if you want to hear Steve here at the studio talking all about SQRL and everything related. So looking forward to that.

Go to GRC.com for everything that Steve is doing. SpinRite, of course, the hard drive recovery and maintenance tool that's amazing, and you can get your copy there. Information about SQRL, if you want to do some reading about that and research about that. Audio and video of this show can be found there. There you go. There's SpinRite. ShieldsUP!. Audio and video of this show can be found there, as well as transcripts of this show. If you rely on transcripts, you can find it only at GRC.com.

And then if you go to our site, it's TWiT.tv/sn. There you're going to see Leo in the banner image, who will be back on this show next week. But you'll also find audio and video of all of the episodes, all the ways to subscribe, all the information that you need just a click away by going to TWiT.tv/sn. We record this show every Tuesday live at 1:30 p.m. Pacific, 4:30 p.m. Eastern. And I checked the UTC today. It's 21:30 UTC now after the time change here in the U.S. So if you want to watch us live, you can, TWiT.tv/live. And you can participate in the chat room, like so many folks today were chatting about the topics that Steve was talking about today.

Steve, excellent job, and I've really enjoyed doing the show with you this past month. Thank you so much for bringing me along.

Steve: Jason, it's been a great four episodes in the past month. And next time Leo wanders off to somewhere, I hope you'll come back for us.

JASON: Absolutely. Any time. I'm happy to do it. I love hanging out with you. Thanks, everyone, for watching this episode, watching or listening to this episode of Security Now! with Steve Gibson. We'll see you all next week with Leo back in the chair. Bye, everybody.

Steve: Bye.