The KNOB Is Broken

Description: This week we look at last week's monthly Patch Tuesday and its collision with third-party AV add-ons. We examine four years of Kaspersky unique web user tracking. We look again at Tavis Ormandy's discovery of the secret undocumented CTF protocol, wondering WTF is CTF? We note a new and devastating strategy in the ransomware battle which hit Texas last Friday. We also have the sad demise of Extended Validation certificates, the further removal of FTP support from web browsers, Google's campaign to still further reduce web certificate lifetimes, and Netflix's discovery of eight implementation flaws in the new HTTP/2 protocol. We'll cover a bit of miscellany, update on my file syncing journey, touch on SQRL news and SpinRite, then conclude with a look at the most recent attack on Bluetooth pairing negotiation which renders all Bluetooth associations vulnerable to a trivial attack.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-728.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-728-lq.mp3

SHOW TEASE: It's time for Security Now!. We end our 14th year of broadcasting and begin our 15th with a banner episode. We will talk about a banner update from Microsoft. How many things did they fix? A bunch on Patch Tuesday. Why you might want to consider dumping your Symantec and Norton antivirus, and maybe Kaspersky, too, while we're at it. Then we'll talk about a new Bluetooth attack called KNOB. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 728, recorded Tuesday, August 20th, 2019: The KNOB Is Broken.

It's time for Security Now!, the show where we cover your security, your privacy, We cover how Internet, computers, technology, networks work. We do it all with this guy right here, Mr. Steve Gibson of the GRC Corporation. Hello, Steven.

Steve Gibson: Hello, my friend. Welcome back. Jason held the fort down while you were having a podcast movement in...

Leo: It came out beautifully, too.

Steve: ...Orlando or something, wherever it was.

Leo: Thank you, Jason, for filling in for me. Yeah, we were in beautiful hot Orlando.

Steve: Ah, yes.

Leo: It was sweaty. But the funny thing is, it's very tropical. They have these massive rainstorms every afternoon. But it was a lot of fun.

Steve: And moments ago I saw the news on, I think it was The Verge, announcing that we were going to have a fourth Matrix movie, that Keanu and - what?

Leo: Matrix 2 and 3 were so darn good, why would you stop there?

Steve: Ah, well.

Leo: I can't say I'm excited about another bad sequel to one of the great movies of all time.

Steve: Yeah, it was one of the great movies of all time.

Leo: Yeah. But it really fell down in 2 and 3, I thought.

Steve: Yeah.

Leo: Oh, well.

Steve: I agree. So, well, Leo, I don't know if you looked at the Picture of the Week, but it kind of gives away what's happening. The very first podcast you and I ever recorded was August 19th - today's the 20th - August 19th of 2005. So I remembered Elaine's correction from a couple months ago, that we would be starting Year 15 on the podcast of August 20th, which is today.

Leo: Oh, Happy Birthday!

Steve: This is the podcast's 14th birthday. We have survived 14 years of escalating security dilemmas and resolutions and interests, and we're plowing now into...

Leo: It's hard to believe.

Steve: ...today Year 15.

Leo: If you had gone to Podcast Movement, the podcast convention, you would have realized how old we are compared to this group of young people who have just discovered podcasts.

Steve: And Leo, we look the same as we did 15 years ago.

Leo: We have not changed in 14...

Steve: Although this moustache seems to be getting a little whiter, so.

Leo: The rest of you looks the same. That's the good news. Keto is keeping you in good shape.

Steve: That's right. We're going to take a look at last week's monthly Patch Tuesday, which was interesting, and its collision with some third-party AV add-ons. We examine four years of Kaspersky's unique web user tracking that nobody knew was going on.

Leo: Can you believe that one? That's so annoying.

Steve: Oh, yeah. We're going to look again at Tavis Ormandy's discovery of the secret undocumented CTF protocol, wondering WTF is CTF? We also note a new and devastating strategy in the ransomware battle which hit Texas last Friday. We've got the sad demise, well, pending, of extended validation certificates. Not looking good for those EV certs. The further removal of FTP support from web browsers. I don't think anybody cares. And of course, if you do, you can just get an FTP client, so come on. We also have Google's campaign to still further reduce web certificate lifetimes that I have some strong opinions about.

Also Netflix, twice now, like I didn't even know Netflix had a security group. Actually, it's the same guy, Looney, who isn't, because he's discovered eight implementation flaws in the new HTTP/2 protocol, which we're going to talk about. We'll also cover a bit of miscellany, a quick update on my file syncing journey, touch on a little SQRL news and SpinRite, and then conclude with a look at the most recent attack. And, boy, I was getting some dj vu because I'm thinking, didn't we already know about this? But I guess not. The most recent attack on Bluetooth pairing negotiation...

Leo: Oh, this one's fun.

Steve: ...which renders all Bluetooth associations vulnerable to a trivial attack known as "the KNOB attack." So I think, yes, another great podcast for our listeners as we start into a robust Year 15.

Leo: A robust 15th year. That's kind of amazing. Happy Birthday. Or Happy Anniversary. I don't know. I guess you can't call it a birthday if it's a podcast. Steverino, let's go on with the show.

Steve: So last Tuesday was another busy and important Patch Tuesday.

Leo: No kidding.

Steve: Of the 93 vulnerabilities Microsoft patched, a third of them, 29, or actually more than a third, are rated critical, and 64 were rated important. Happily, for a change, none of the vulnerabilities patched last Tuesday were known to be under active attack, nor had any of their details been published publicly. I did see a tweet from someone, and I attempted to follow up, but I couldn't, saying that SandboxEscaper was tweeting again yesterday. But I looked, and there wasn't anything on her GitHub account, and I didn't further track her down. So I don't know what it was she was tweeting about. But because of course she's been responsible for lots of zero days recently. The two previous Patch Tuesdays, what were patched in some cases were zero days being exploited at the time. So in this case no.

But there was still a bunch of stuff. There were four remote code execution bugs in Microsoft's recently troubled remote desktop system. Of course we talked about the RDP protocol and how with BlueKeep, you know, everybody was expecting a worm. I wasn't because it was so easy to exploit, you really didn't need a worm's automation to help you. You could just go log into somebody else's Bluetooth system. So unlike the EXIM email server vulnerability, where it took a week of dribbling out bytes in order to hold the connection long enough for that weird undelivered mail timeout to occur, and then you're able to exploit a problem with the server, in this case you just say, hey, I want to connect you. And thanks to this vulnerability, the BlueKeep vulnerability, that was possible.

Now we have four more, two of which are of big concern. However, these are not in the Remote Desktop Protocol. They're in sort of the higher level enterprise version, so-called "terminal services." So that's RDS, the Remote Desktop Services. That's what enterprises use when people want to actually log into a window instance through terminal services. It's like the professional enterprise version of Remote Desktop. With Remote Desktop, you can connect to your one Windows instance. But, for example, you're logged off of the desktop if you try to log in through Remote Desktop because Microsoft says, oh, no no no no, only one person can be logged into your consumer Windows at a time. So that happens.

Anyway, so those are being - four of those things were found by Microsoft themselves. So they found the problems internally when they were taking a closer look at Remote Desktop in general, no doubt motivated by BlueKeep, thinking, hmm, we haven't looked at Remote Desktop for a decade. Maybe we should take a look at it again and see if there's something else we need to fix. And they found four remote code execution problems, two of which were also wormable. Again, not known to be happening out in the wild in this case. And those got fixed last week, a week ago.

And beyond that, there are seven other remote code execution problems which impacted the Chakra scripting engine. That's the original scripting engine in Edge and is used in some Microsoft apps. Of course they'll be moving away from that as Edge goes over to Chromium. There were also two remote code execution vulnerabilities in Microsoft's Hyper-V virtual machine technology, six remote code executions in Microsoft Graphics component, one in Outlook, two in Word, two in the Windows DHCP client - and you don't want that because that's heavily networked. And also two in the older Scripting Engine component. Oh, and one in the VBScript engine.

And I also noticed some dialogue somewhere, it didn't make it into the podcast formally, but just that VBScript was really being sunset at Microsoft. I mean, they, of course, when JavaScript was happening, they're like, oh, well, let's see, what do we have? Oh, we have Visual Basic. Let's turn that into a scripting language and so people can write Visual Basic script for their browser. And people did. And of course it lost the battle to what is now formally ECMAScript, which is the formalization of browser-side scripting, which has now become fully standardized.

There's also a patch for a vulnerability in this, as I mentioned, this CTF protocol. Just not clear how they're going to patch this because this is so badly damaged. And even Tavis is kind of scratching his head, thinking, okay. Anyway, this CTF protocol impacts all versions of Windows since XP, when this was mysteriously introduced. And we'll talk about that in a minute further.

So overall, this month's August 2019 Patch Tuesday is large and important. Oh, and Microsoft also wanted to remind users in one of their "what's happening this month" notices, they just said, oh, by the way, don't forget, Windows 7 and Windows Server 2008 R2 will be out of extended support and no longer receiving - and they didn't say it, but I'll just say "free updates," because we know that enterprise users will be able to pay an escalating, please keep us on the IV drip for, what is it, one, two, or three years continuing.

And I'm really interested to see whether Microsoft holds to this plan. I mean, we know how desperately they are working to get people off of 7 and Windows Server 2008 R2, which is the server version of Windows 7. But it's still neck and neck with Windows 10. It was at the beginning of the year that they switched places in terms of who had the lead, but 7 hasn't been dropping that much. I mean, some, as the curtain is trying to fall on it. But again, we've seen Microsoft say, oh, maybe we'll - well, I mean, they've done it with Windows 7. We're now in the extended support period which they just decided to give to everybody.

So we'll see what happens. That's going to be interesting because they just, you know, what'll happen, I think, is that over time machines will die, and you can't get a new machine to run Windows 7 any longer. Windows 7 won't install with USB3 support, that is, its installer doesn't know about USB3. So you have to jump through some hoops, and I've jumped through those hoops, in order to get Windows 7 to set up on a machine with USB3, which all machines now and for quite some time have had. So it's, you know, what'll happen is I think these machines will just die, and they'll get replaced with machines that have Windows 10 for better or for worse. But I encounter people, you know, like in the real world all the time that just hate Windows 10. And so we'll see what happens.

Also Adobe, SAP, and VMware had respective Patch Tuesdays last week. Adobe published fixes for Photoshop, Experience Manager, Acrobat and Reader, Creative Cloud desktop app, Prelude, Premiere Pro, Character Animator, and After Effects. And no Flash security updates this month. Maybe it's just because people have, you know, it is certainly, although it's kind of around, it's certainly of diminishing impact. There are much larger targets to attack these days, like all versions of Windows since XP in the case of this CTF exploit that we'll talk about in a second.

But what was interesting was there was a problem last Tuesday. Recall how we talked about this at the beginning of the year. Microsoft announced that they are going to stop cosigning their Windows Update packages which have for quite a while been signed with both SHA-1 and SHA-256 hashes. They decided that July would be the last month where the cosigning would occur, and that after July, meaning starting in August this month, that is to say last Tuesday, updates would only be signed with SHA-256. And that makes sense because it made sense to sort of straddle the hashes. But at some point, if you really do believe that the weaker of the two, that is, SHA-1, isn't trustworthy, then that's the one you need to remove; right? Because if you cosign, your signing is only as strong as the weaker of the two signatures, if you're going to accept either one. So they had to get rid of it.

So a week ago things broke, although not what Microsoft or anyone else was expecting. Remember that there was an update to Windows which, at the time it was first announced, it wasn't clear whether they were going to push it out automatically because

the wording in their update announcement sort of made it sound like you had to go get it. Like you were going to be in trouble if you didn't go get it. But no. They did push it out. And so people would have updated. And that was the update which mostly affected Windows 7, because it didn't know about SHA-256 back in the day, which would be required for any updates to be accepted from August on.

Well, it turns out that Symantec and Norton third-party AV tools have been making it their business to protect users from unsigned or mal-signed Windows updates, which is kind of weird since that's the whole reason they're signed in the first place, so that Windows can and will and does already robustly protect itself from any and all possibly weird or unsigned updates. I mean, it's like, it's not going to accept them. But I guess in this, like, well, what can we do to justify our existence more, Symantec and Norton said let's check those.

So apparently the Symantec and Norton AV update protectors did not get the memo about the signing changing on Windows Update, with SHA-1 dropping last month. They were both only checking older SHA-1 signatures. Which of course broke last week. Both of those AV systems refused to allow their client machines to receive any of Microsoft's valid Patch Tuesday updates. Whoopsie.

There's little that Microsoft can do at this point. That stuff, as we know, is installed in users' machines, and it's blocking Microsoft from doing anything. So it knows, Microsoft Update does know about the inventories of the machines it's updating. So Microsoft has updated Windows Update to stop attempting to send any of this month's, last week's, updates to any machines containing Symantec or Norton AV. And as a consequence...

Leo: Geez, that's a real secure situation.

Steve: Isn't that a mess, Leo?

Leo: Oh, my god.

Steve: It's just unbelievable.

Leo: Oh, you've got an antivirus? By the way, those are the two most common and most popular and often included as trials on new machines. So it's highly likely that a vast number of Windows users already have it. Which means they're not getting updates?

Steve: Now, get a load of what Symantec says. It's just the most mealy response. Symantec wrote: "Symantec has identified the potential" - uh-huh - "for a negative interaction between Symantec Endpoint Protection and the contents of future Windows Updates" - okay, current and future - "as a result of the changes in this Microsoft KB," you know, Knowledge Base. "Out of an abundance of caution," Leo...

Leo: Yes?

Steve: Uh-huh, "Symantec and Microsoft worked together to only allow the update to be visible to versions of Symantec Endpoint Protection that fully support SHA-2 signed

Windows executables replaced by this and future updates to Windows 7 SP1 and Windows 2008 R2 SP1."

Leo: What? I don't understand what that just said. You just said it. I heard it. It was in English. But what does it mean?

Steve: It said nothing.

Leo: Are they going to fix their problem?

Steve: Yeah, Symantec is - oh, Leo, here's the good news.

Leo: Oh.

Steve: "Symantec is actively working on multiple releases" - because this affects everything - "multiple releases of Symantec Endpoint Protection to address this situation." Which of course what you just read explains nothing about.

Leo: Yeah.

Steve: "This document will be updated." Whew. We'd just rather have the software updated, but okay.

Leo: Oh, please.

Steve: "[This document] will be updated as each release becomes available for distribution and include details on how each update can be acquired." So, yeah. I'm glad you and I are not using any of this nonsense.

Leo: Oh, what garbage.

Steve: I know.

Leo: So it's unconscionable that an antivirus would block Windows updates, period.

Steve: Correct.

Leo: I guess they're not, it's just they're using SHA-1, which is broken. And we've known that for how long?

Steve: Well, and that's the point is that, well, we've known that all year, that it was going to - that SHA-1 was...

Leo: This was coming.

Steve: ...was going away at the end of July; that everything from August on. So they're blocking the signatures. And again, they're not offering any benefit by doing this because Windows won't install an update that it hasn't verified. So they're just, like, one additional monkey in the works. Or wrench in the monkey or something, I don't know. There's definitely a wrench somewhere. It's just amazing.

Leo: Wow. By the way, Symantec and Norton I think are the same companies. I know they are.

Steve: Right, someone bought the other.

Leo: Symantec bought Norton. I think they rebranded older Norton stuff. No, maybe not. I think they still use Norton for the consumer brand and Symantec for the business brand.

Steve: Ah.

Leo: And I think Symantec Endpoint is business. So anyway, this is terrible. That's not what a security software should do.

Steve: Right. So you install this protection, and it prevents you from obtaining your monthly IV drip of security patches. It's just incredible. So anyway, for our listeners who are using Symantec or Norton, you keep an eye out. I mean, I'm sure that those packages will - and I checked. The link in that note was current as of yesterday, where they said, oh, uh, we're working on it, and we'll update this as updates become available. Well, there were none available yesterday.

So we're now, you know, thank goodness that none of the things that Microsoft fixed are, like, being exploited in the wild, or you'd really be in much worse trouble. So I'm sure that Symantec and Norton will update themselves, and then I don't know what. Then maybe users, if you see that, you should go and check for updates. I mean, I'm sure Windows will get around to pushing those out again. It's just, as you said, Leo, it's a mess.

Leo: Ugh. Ugh. Ugh.

Steve: And in another foul-up this month, Microsoft, in that same note where they talked about this problem, speaking of one of the patches from last week's updates: "After installing this update, applications that were made using VB6 (Visual Basic 6), macros using Visual Basic for Applications, and scripts or apps using Visual Basic Scripting Edition (VBScript), may stop responding" - and actually it's one of those, again, will stop responding - "and you may [and you will] receive an 'invalid procedure call error.'"

They said: "This issue is resolved in KB4517297, which is an optional update." In other words, last Tuesday they broke Visual Basic and only found out after the fact and said whoops and so have now got an update to their update which unbreaks it. So there's that.

And while we're on the topic of AV that nobody wants, Kaspersky, it turns out, has for four years, since late 2015, been facilitating independent web tracking of every single one of their users. Both the free and the paid editions have been injecting a snippet of JavaScript into every page, every web page displayed by their users. The JavaScript, and I've got a picture of it in the show notes, which is placed into the page's DOM, the Document Object Model, can therefore be readily seen by and parsed by any other script running on the page. That's the whole point of having a Document Object Model is we've standardized the way pages are described.

And anybody looking at this little bit of script - you've got it on the screen right now, Leo - you can see that it's a script calling out the downloading of, over HTTPS from a server, gc.kis.v2.scr.kaspersky-labs.com. Then we have forward slash, then a GUID, you know, a Globally Unique ID, then /main.js, and then close script. So this little bit of code causes the user's browser to go download main.js from Kaspersky.

Well, this GUID, this Globally Unique ID, is the ID from the individual user's instance of Kaspersky. And that GUID is inserted into every page downloaded. And that GUID can be seen by any script running on the page, including advertisers. Meaning that, since late 2015, any user of Kaspersky has been having it embed their unique ID in every single page and every single ad that they download, making them absolutely trackable. I mean, you can't shake this. It's just there.

Not to mention the fact that Kaspersky themselves, since every single page you download is fetching this main.js from Kaspersky Labs' servers with this unique user tag in the query, and we know that the browser will provide a referrer header telling Kaspersky exactly where and what page who you are is on, they've had the ability to globally track their entire install base everywhere they go on the Internet. We don't know that they were, but that's what this does. And so they certainly could have been. Until it was made public two months ago, when they said, oh, okay, we won't do that anymore.

So they've changed it to a new ID, which is no longer user unique. It's now Kaspersky version unique. So now what's being transmitted, it's a much less probably source of concern hopefully, is the fact that you're a Kaspersky user, and exactly which version of Kaspersky AV you are using. And Leo, I am so glad that neither of us have any of that crap on our systems.

Leo: Yeah, no kidding.

Steve: Thank you anyway.

Leo: That's just bad behavior. And as somebody's pointing out, I mean, you're announcing at the very least that you're using this antivirus.

Steve: Right.

Leo: So, I mean, just - yeah. Unbelievable. Unbelievable.

Steve: So what the heck is CTF? When I was talking about Windows updates, I mentioned that there had been a patch to it. So what is it? Once again, Google's Tavis Ormandy discovered this buggy protocol which, if hackers or malware - and this has been around since Windows XP, when it was introduced. If hackers or malware had already gained a tentative foothold on a user's computer - so this is not a remote vulnerability, this is any Windows app running on anyone's machine - they can use it to take over any app, high-privileged applications, or the entire OS as a whole, Tavis explains.

It turns out that CTF, which no one has ever heard of before, is a little-known Microsoft protocol used by all Windows operating systems since XP, right up through 10. And naturally it's insecure, and turns out can be easily exploited. So what is CTF? Nobody knows for sure. We don't know what CTF stands for.

Leo: Capture The Flag?

Steve: Love it. Perfect.

Leo: Yeah, yeah.

Steve: Tavis was never able to determine what it means, despite rummaging through all of Microsoft's documentation.

Leo: Continuous Thermonuclear Fusion?

Steve: That'd be good.

Leo: Yeah, yeah.

Steve: What Tavis did learn...

Leo: It's all from the chatroom.

Steve: Thank you, chatroom - that CTF is part of the Windows Text Services Framework, so "T" of CTF might be Text. We don't know.

Leo: Casper the Friendly Protocol?

Steve: There is a TSF, the Text Services Framework. So this is the system that manages the text shown inside Windows and Windows apps, that is, Windows itself and Windows applications running on Windows. When users start an app, Windows also starts a CTF client for that app.

Leo: Interesting.

Steve: The CTF client receives instructions from a CTF server about the OS system language and the keyboard input methods, so it's tied in to Text Services and multi...

Leo: I wonder if it's a clipboard function of some kind. Clipboard. Right?

Steve: "C" could be clipboard, yeah. Clipboard Text Facilitator. Anyway.

Leo: Yeah, something like that, yeah.

Steve: If the OS input method changes from one language to another, then the CTF server notifies all CTF clients, who then change the language in each Windows app accordingly in real time. So this whole thing was a hack to allow you to, on the fly, change the language of the system, and suddenly everything would go gabloop and change.

Leo: We should send this Microsoft note to Tavis. It's the Collaborative Translation Framework.

Steve: Ah, interesting. I mean, it's funny that at the time of his reporting he was unable to determine what the heck it was.

Leo: Yeah, yeah.

Steve: So Tavis dug into it, and he quickly discovered - to unfortunately not surprisingly, and not, well, yeah, not surprisingly because it's he - to his shock and dismay that the communications between CTF clients and the CTF servers are not properly authenticated or secured. Or, as Tavis put it: "There is no access control in CTF."

Leo: Yeah, because the point of it - by the way, it's been deprecated - was collaborative translations. So you can submit a sentence translation. Others can submit. And then it collaborates based on - it returns the translated content in its total count from your account. Yeah, it's an API for collaborative translation. So it makes sense. It is a server.

Steve: And it's deprecated but still present because, you know, five people are using it somewhere.

Leo: Right, yeah, you can't kill it. It was only deprecated last February 2018, so yeah.

Steve: Yeah. Any application, any user, even sandboxed processes can connect to any CTF session. Clients are expected to report their thread ID, their process ID, and their main Windows messaging handle. There's no authentication involved. And this is Tavis: "No authentication involved, and you can simply lie." Tavis added: "So you could connect to another user's active session" - I mean, this is complete violation of all containment

within Windows - "connect to another user's active session and take over any application, or wait for an administrator to log in and compromise their session."

An attacker that hijacks another app's CTF session can then send commands to that app, posing as the server, normally expected to be the Windows OS, but there's no enforcement of that. Attackers can use this loophole to either steal data from other apps, or they can use it to issue commands in the name of those apps. If the apps run with high privileges, then those actions allow the attacker to take full control over a victim's computer. And, according to Tavis, any app or Windows process is up for grabs. Because of CTF's role to show text inside any app or service, there is a CTF session for literally everything and every user interface element on Windows OS.

To demonstrate the dangers, Tavis recorded a demo in which he hijacked, get this, the CTF session of the Windows login screen which, as we know, Microsoft has made a huge deal about being highly privileged, isolated, and sacrosanct. Thus Tavis easily demonstrated that everything in Windows is hackable because of CTF. I have a link to the Google Project Zero Blogspot posting. Tavis titled it "Down the Rabbit Hole." And he said toward the end, he concluded, he says: "I've implemented this attack in ctftool. Follow the steps here to try it." And it's on GitHub, a link to ctftool, where it shows you how to achieve this.

And he concludes: "So what does it all mean? Even without bugs" - and there are bugs. But "Even without bugs, the CTF protocol allows applications to exchange input and read each other's content. However, there are a lot of protocol bugs that allow taking complete control of almost any other application. It will be interesting," he's writing, "to see how Microsoft decides to modernize the protocol." He says: "If you want to investigate further, I'm releasing the tool I developed for this project."

And then, under conclusion: "It took a lot of effort and research to reach the point that I could understand enough of CTF to realize it's broken. These are the kind of hidden attack surfaces where bugs last for years. It turns out it was possible to reach across sessions and violate NT security boundaries for nearly 20 years, and nobody noticed." So Tavis...

Leo: Wow.

Steve: Yeah. Thank you, once again. I mean, you know, anybody who's programmed Windows recognizes how much danger the whole Windows messaging model creates. I mean, you are able to enumerate processes. You are able to get the messaging queues of other apps. I mean, this is what macro recorders and players do is they send keystrokes to apps, causing them to do things. And it's nice when it's your macros that are being sent, and your apps, and they're doing what you want them to do. But nothing enforces that in Windows. I mean, so essentially you really need to make sure nothing gets in because once something has, it's game over.

Leo: Unbelievable.

Steve: So the news, as of Friday, from Texas was government agencies, nobody knew what government agencies, but we knew there were 23 of them, were all hit with a - well, now we know it's cities, we'll get to that in a minute - were all simultaneously hit with a well-coordinated, simultaneous, and effective ransomware attack last Friday, August 16th. Twenty-three Texas entities, the majority of which are local governments,

were hit by a ransomware attack on Friday that Texas officials say is part of a targeted attack launched by a single threat actor.

Details still remain scant about the specific agencies hit by the ransomware attacks, which began on the morning of August 16th, as well as exactly which systems were impacted. However, the Texas Department of Information Resources, DIR, as of Saturday night did say that responders are actively working with all 23 entities. And who knows how many were attacked that didn't get infected. But 23 different things, different networks, different agencies, now we know it's actually 23 cities were brought down by a massive coordinated attack.

So this DIR, the Department of Information Resources is currently working, as of Saturday night, to bring their systems back online; and the state of Texas systems and networks itself were not impacted. So what we do know is that these 23 agencies were knocked offline and presumably encrypted by this simultaneous attack. I checked yesterday in local reporting, like I don't know, the Texas Gazette or something, and it didn't have any more information still.

The Texas Department of Information Resources website posted a statement saying that, quote: "Currently, DIR, the Texas Military Department, and Texas A&M University System's Cyber-Response and Security Operations Center teams are deploying resources to the most critically impacted jurisdictions. Further resources will be deployed as they are requested." When pressed for additional details, the Texas DIR declined to elucidate any further, stating "due to security concerns" - read embarrassment, perhaps - saying only that they were smaller local governments. Yes, 23 of them.

The DIR did not provide information about which systems are down, how systems were first infected, and the specific amount of ransom. You know that's going to be adding up. In their reporting of this, Threatpost reached out to representatives from Dallas, Houston, and Austin for comment on whether they were impacted by the attack. While representatives from Dallas and Austin have not yet responded - I wonder if their email's down - a spokesperson from Houston told Threatpost that: "As far as we know, Houston has not been affected." I think they would know if they had been.

According to a statement sent to Threatpost: "The city of Houston is aware that a ransomware attack has affected several local government agencies throughout Texas. We are in contact with the Texas State Operations Center and will monitor the latest developments." Whew. "The Mayor's Office of Homeland Security and the IT Services Department will continue to proactively work to secure and protect the city's assets." However, the DIR said that at this time, and not surprisingly given what we know, evidence gathering indicates the attacks came from one single threat actor.

Allan Liska, who is a threat intel analyst with Recorded Future - we've spoken of them a number of times, they're an in the middle of the thick of things security firm - told Threatpost that the attacks signify an important shift in the ransomware threat model. "Typically, state and local governments have been targets of opportunity for ransomware attacks, with the gangs behind these attacks, Ryuk and SamSam, appearing to stumble onto previous state and local government targets." Or, that is, stumble previously onto them. "However, this incident," he says, "appears to be the first where a string of governments were actively being targeted in an attack."

He said: "This is the first time there's been an attack against several local governments in a state." He says: "This is big. It's a game-changer. This will change the model going forward for attackers, and that will be a problem for governments." Allan also noted that one advantage Texas has is it has a coordinated incident response. The response team is centralized for cities and counties in emergencies, which makes it easier when there is a problem like this to find and reach out to a main contact. There's someone to call. So

anyway, as I said, I looked for any update as I was putting these notes together yesterday, and everyone is being quiet. They're now saying that it's 23 individual cities across Texas. But that's all we know.

And I shudder to think of the ransom that is being requested. I mean, say there are some cities that do have current backups. In our previous reporting of this over the last few months, we've noted that recovering from an attack like this is sometimes more, like without paying ransom, is sometimes more expensive than if you pay the ransom and you are able to decrypt the systems in place, just due to the logistics of having systems which have been there, who knows how old they are, what software they have on them, how recent the backups are, whether the backups were online and therefore themselves also got encrypted in the process. It's a mess. I mean, this really is an interesting, really significant new problem for U.S. municipalities. I mean, it's just not exaggerating to say that.

We have, unfortunately - because, I mean, I guess I understand this, but it's still unfortunate - the coming demise of Extended Validation (EV) certificates. Safari has already removed all EV certificate company info from the address bar, where it had been before this most recent major update to iOS and macOS. Most mobile browsers are not showing it because they have scant real estate, and they just don't want to take the space for it. Now both Chrome and Firefox browsers for the desktop have announced that they, too, will soon be removing any EV indication from the main URL UI.

Chrome's Google Groups post was titled "Upcoming Change to Chrome's Identity Indicators." And it reads: "As part of a series of data-driven changes to Chrome's security indicators, the Chrome Security UX [User Experience] team is announcing a change to the Extended Validation certificate indicator on certain websites starting in Chrome 77. On HTTPS websites using EV certificates, Chrome currently displays an EV badge, containing the name of the EV certificate holder" - you know, like mine says Gibson Research Corp. - "to the left of the URL bar. Starting with Version 77, Chrome will move this UI indicator down into the Page Info" - which of course you have to click on in order to see, meaning it just might as well not exist, most people don't even know it's there - "which is accessed," they say, "by clicking the lock icon." In other words, effectively nullifying its impact.

They said: "Through our own research" - and again, this is where I get the, yeah, I understand. "Through our own research, as well as a survey of prior academic work, the Chrome Security UX team has determined that the EV UI does not protect users as intended. Users do not appear to make secure choices, such as not entering a password or credit card information, when the UI is altered or removed" - meaning actually they never knew what it meant, and maybe they wouldn't have cared even if they did - "as would be necessary," they write, "for EV UI to provide meaningful protection. Further, the EV badge takes up valuable real estate, can present actively confusing company names in prominent UI, and interferes with Chrome's product direction towards a neutral, rather than a positive, display for secure connections."

And I'll intervene here just a moment to say, in other words, as we know, secure is intended to be the norm now going forward, and non-secure sites are what will then be denigrated for, like, something - as we know, the UIs will begin saying "not secure" proactively; whereas, if they don't say that, then it's secure. And that will then just be the de facto.

Anyway, they said: "Because of these problems and its limited utility, we believe it belongs better in Page Info," meaning if somebody cares. And of course, who does? "Altering the EV UI is part of a wider trend among browsers to improve their Security UI surfaces in light of recent advances in understanding of this problem space." They write:

"In 2018, Apple announced a similar change to Safari that coincided with the release of iOS 12 and macOS 10.14 and has been implemented as such ever since."

And shortly following Chrome's announcement, Mozilla also announced that, starting in Firefox 70, they will be removing the EV certificate's identity information from the address bar. Mozilla wrote: "In desktop Firefox 70, we intend to remove Extended Validation (EV) indicators from the identity block, the left-hand side of the URL bar which is used to display security/privacy information." Actually, mine's - I'm noticing mine's getting kind of full of little icon-y things. I had, like, five of them this morning when I was looking at something, you know, the half shield and all kinds of stuff.

They said: "We will add additional EV information to the identity panel" - uh-huh, thank you - "instead, effectively reducing the exposure of EV information to users while keeping it easily accessible." And once again, yes, where nobody will notice. They wrote: "The effectiveness of EV has been called into question numerous times over the last few years. There are serious doubts whether users notice the absence of positive security indicators, and proof of concepts have been pitting EV against domains for phishing. More recently, it's been shown that EV certificates with colliding entity names can be generated by choosing a different jurisdiction."

What that means, for anyone who's interested, is, for example, it would be possible, I can't remember the example that was given, but for example it would be possible for a Gibson Research Corp. to be incorporated out of California, because I'm a California Corp., in some other state. Nothing prevents that. And that's their real name. They can't get GRC.com, but they could get something, you know, and get an EV certificate for their real name. And so it would show Gibson Research Corp., just as mine does, and it would be a valid EV certificate. So that's what both Google and Mozilla mean when they say, you know, it's not really clear what it means to show the company name because you can have multiple companies with the same name.

Then they said, "Eighteen months have passed since then, and no changes to address this problem have been identified." They said: "The Chrome team recently removed EV indicators from the URL bar in Canary and announced their intent to ship this change in Chrome 77. Safari is also no longer showing the EV entity name instead of the domain name in their URL bar, distinguishing EV only by the green color. Edge is also no longer showing EV entity name in their URL bar." So RIP, EV.

I think that pretty much, you know, given that EV certificates are significantly more expensive for website domains to obtain - I read a bunch of other reporting of this that wasn't necessary to put into this relative long summary. But, for example, it is substantially more difficult for an EV certificate holder to prove and need to continuously update and re-prove their identity. You know, I do it because it was a badge of honor to have an EV showing. I wanted to. That's going away. So I think, you know, it's unfortunate because I'll be talking in a minute about Google's efforts to work to shorten certificate lifetimes rather than to deal with revocation.

So as Chrome moves forward, things change. They have posted their intent to remove and deprecate FTP support. And again, it's like, okay, fine.

Leo: How many people really use your browser for FTP? If you're serious...

Steve: I kind of forgot it was still there. You know, when have you have seen an FTP URL?

Leo: Ftp://; right?

Steve: Yes, exactly. Yes. So they said: "The current implementation in Google Chrome has no support for encrypted connections, FTPS, nor proxies. Usage of FTP in the browser is sufficiently low that it is no longer viable to invest in improving the existing FTP client. In addition, more capable FTP clients are available on all affected platforms. Google Chrome 72 and on removed support for fetching document subresources over FTP and rendering of top-level FTP resources. Currently, navigating to FTP URLs result in showing a directory listing or a download, depending on the type of resource." In other words, the document subresource, meaning like components of documents.

So from Chrome 72 you could not have a web page that might have, for example, been loading a picture. I don't know why you would. But normally it's https:// and then the URL to .jpg. Well, you could have had that picture on an FTP server, and in the document, in the page it would have said ftp://. But anyway, so that was removed from Chrome 72 onward. Those are document subresources. But you could still, in the URL bar, you could put ftp://, you know, ftpforever.com, and up would come the directory of that root, that directory of the FTP server.

So they said: "A bug in Google Chrome 74 resulted in dropping support for accessing FTP URLs over HTTP proxies. Proxy support for FTP was removed entirely in Google Chrome 76." So rather, you know, there was a bug. They said let's just kill it. Let's not fix it. No one cares.

Finally, they said: "Remaining capabilities of Google Chrome's FTP implementation are restricted to either displaying a directory listing or downloading a resource over unencrypted connections. We would like to deprecate and remove this remaining functionality rather than maintain an insecure FTP implementation. So to that end, Chrome will soon be getting a new DisableFTP flag which will initially not be enabled at first," meaning that DisableFTP won't be active.

So this remaining shred of FTP functionality will still be present. But over time we can expect them to migrate that away so that it will eventually be enabled. Then that remaining bit of FTP won't be enabled by default. Those who do want it will be able to turn it on - I'm sorry, will be able to turn off the Disable, thus enabling it. But that will really be a clue that they ought to go find themselves another client. So anyway, FTP is going, and no one is going to miss it.

Unlike shorter certificate lifespans. Two months ago, in June, Google's Ryan Sleevi, who's a good guy, by the way, introduced a ballot measure during the CAB conference proposing to amend one of the documents, removing lines and adding lines, to reduce the maximum browser security certificate lifetime to essentially one year, you know, one year plus a little bit of fudge so that you don't have a problem with overlap. Basically a year. Right now EV certificates are two; OV certificates are three. So for those using OV certificates, the more popular certificate, that triples the rate at which certificates must be reissued, which is annoying because the crypto is secure. I mean, there's no reason. Well, okay, except there is kind of one.

Anyway, the measure was titled "Ballot SCXX." I love this. "Improve Certificate Lifetimes." That's right, it's an improvement. Just the fact that Google said "improve certificate lifetimes" when "reduce certificate lifetimes" is what they really mean demonstrates that they clearly recognize that certificate revocation is still broken. My first thought upon seeing that this was in the news was of course they are, since as our longtime listeners of this podcast know, that is, those who have been listening for at least five years, Google's Chrome browser certificate revocation is not completely and utterly broken; rather, it is nonexistent for all practical purposes. Chrome doesn't even

bother checking non-EV certificates with their proprietary CRL - that was supposed to be Certificate Revocation List - CRLSET system. And now of course they're working to kill EV certificates, too, so they won't even need to be checking those in the future.

For some unfathomable reason, back in the dawn of Chrome, someone must have made the decision that, since certificate revocation was currently imperfect, it was therefore of no value at all. So unlike every other browser on the planet, Chrome chose to simply ignore it. It was, at the time, I think, an irresponsible and unconscionable choice. Oh, sure. They have their own web browser rigorously checking any and all of their own properties' certificates. You don't dare mess with a Google-derived cert, as we have often noted. But they don't bother to check anyone else's, only their own.

When I clearly demonstrated this five years ago, back in May of 2014, by deliberately creating a revoked certificate, which Chrome gleefully accepted, I mean, it was revoked. Chrome said, eh, fine. The other browsers said, no, wait, this is revoked. No, you can't go there. We won't show you that page. Chrome said, yeah, come on in. What happened? Google manually added that one certificate's signature to Chrome's short certificate blacklist. So I created another, which was once again honored because, you know, why not? And that one they didn't bother blacklisting. I haven't looked recently.

So I'm sure that this, and I, annoyed them by bringing attention to this original sin of Chrome's. It wasn't my intention to annoy them, but I did hope that by bringing this to light we might see this fixed, since back when Chrome was becoming more and more popular and influential, it mattered. And of course it matters now. All of the other browsers were doing the right thing. And again, yes, the existing system was imperfect. But it made more sense then, and it still does now, to fix it, rather than just simply ignore it.

I haven't revisited any of this since then. So that's been a little over five years. But anyone who is interested in that research and coverage can find those original pages at GRC.com/revocation. And when I was working on this back then, doing that research, I encountered the perfect solution to the problem, which is known as OCSP stapling. OCSP stands for Online Certificate Status Protocol. Certificates contain the URL of their issuer's OCSP server, which allows the current instantaneous status of the revocation state of the certificate to be verified. This enables web browsers to query certificates' OCSP servers to receive a completely up-to-the-instant check on the current validity of the certificate.

The trouble is this has been an emerging standard. And in the beginning OCSP servers could not always be counted on to reply quickly, if at all. So this either slowed things down while the browsers waited - something browsers really hate to do - or browsers were forced to take the position of trusting unless denied. In other words, failing open rather than failing closed. But it turns out a perfect solution does and has existed for years. It's called OCSP stapling.

When OCSP stapling is used, the web server that's offering and asserting the validity of the certificate, it itself obtains and caches an updated and signed assertion of the current validity of the certificate and then staples that to the certificate, which it offers to the web browser. The web browser checks the signature of the certificate and of the recent OCSP validity assertion. So think of the power this gives certificate authorities. Now they can revoke a certificate, and it will become untrusted instantly, and with zero overhead introduced by browsers, and no browser delay.

In fact, two years ago, in July of 2017, Cloudflare's Nick Sullivan blogged with the title: "High-reliability OCSP stapling and why it matters." And I won't go through the whole posting because it's very lengthy, but interesting. I have the link to it in the show notes. But he began, to introduce the concept: "At Cloudflare our focus is making the Internet faster and more secure. Today we are announcing a new enhancement to our HTTPS

service: high-reliability OCSP stapling. This feature is a step towards enabling an important security feature on the web: certificate revocation checking. Reliable OCSP stapling also improves connection times by up to 30% in some cases. In this post, we'll explore the importance of certificate revocation checking in HTTPS" - which again, just to rub it in, which Chrome doesn't bother to do anyway - "the challenges involved in making it reliable, and how we built a robust OCSP stapling service." Anyway, for anyone who's interested, the link is there.

So today, OCSP stapling is universally available. Windows Server supports it, Apache, Nginx, all the various CDNs, AWS and other web providers. So there is no longer any good reason not to simply make its support mandatory in the future. Incredibly, as I've said, Chrome doesn't perform any useful revocation checking. So to minimize the exposure to revoked certificates, languishing for their current up-to-three-year period, their solution is to shorten the certificate's maximum life. They want, Google wants, all certificates to die more rapidly through their natural self-expiration. But of course this still leaves us with a big mess and a gaping hole for exploitation because you still have a year.

You know, it really feels as though Google is working to incrementally chip away at the certificate authority model, with the aim of eventually bringing us to this ACME protocol, where we know nothing about a certificate. Nothing. Where the certificate essentially makes no assertion other than the fact that one automated web server requested and received a domain validating certificate from some ACME server. Essentially, all it then gives you is encryption. It gives you privacy, but it gives you nothing in terms of who the certificate is being issued for because it's now just an automated handshake. And you know, that doesn't feel like the right future for the Internet.

Moving forward, it seems so clear that we're going to need more assurance, more than just encryption. We need to know who it is we're talking to for our connections on the Internet; right? And this is going in the other direction. We're going to really need to know who the server is and have some reason to trust the entity that's there, rather than a bot that issued the certificate in response to an API request. It seems to me this is the vital service that certificate authorities have always provided. It's not perfect, no. But neither was OCSP, and that has been fixed completely with stapling. There's just been no pressure to implement OCSP stapling.

So it seems to me that, rather than discarding certificate authorities, which seems to be the direction that Chrome is wanting to go because they're not doing any revocation checking, they're wanting to just chip away at certificate life, that certificate authorities provide this vital service of having done some due diligence about who it is the certificate is for.

Now, we've talked about alternatives. Someday the DANE protocol, the DNS-based Authentication of Named Entities, that might happen. But DNS needs to be made significantly stronger with DNSSEC, which would need to be universally supported before that could happen. So it seems to me that the much more practical proximate solution is to move OCSP stapling, which is done, it exists, it's ready, we ought to move that to the forefront by visually rewarding in the browser's UI those sites whose certificates carry a freshly stapled OCSP assertion. That would give sites an incentive to implement readily available OCSP stapling.

Basically, they just have to turn it on, which would give us true zero-overhead certificate revocation checking to solve the revocation problem once and for all. And then, yes, we need to figure out how to strengthen the assertion that a manually certificate authority issued certificate is able to make. But it seems to me going forward that it's crazy to imagine an Internet where you've got secure connections, and you have no idea who

you're connecting to. I just - that makes no sense to me. So, but it'll be interesting to see how this evolves in the future.

Netflix, as I mentioned, has been providing some security oversight, which the first time I saw it - there was something else a few months ago where I said, wait, what? Netflix? But yeah. So this is the second time this is happening. They took a look at the HTTP/2 protocol, some actual implementations. For a long time we've had HTTP/1.1. And we've talked about HTTP/2's many new features previously. And just to give our listeners a quick short summary, whereas a single HTTP/1.1 connection makes a request and waits for a reply, then it might make another request during the same connection, or it might terminate it and open another.

With HTTP/2, what we have is an inherently powerful, very powerful multiplex connection supporting multiple simultaneous overlapping streams which support multiple simultaneous overlapping queries and responses, differing stream priorities, and even the ability for the server to see what things are being requested and to anticipate future not-yet-requested by the client assets and send them ahead, even before they're being requested. I mean, it really does feel over-engineered, but it's arguably what today's massive web pages require. And you could imagine some, I mean, these features are present, even though they're not even being used yet. I don't even know of a server that is heuristically analyzing the requests and looking at previous requests of the same stuff and noticing what replies were always being followed up, and then saying, oh, well, if you ask for this, you're probably going to be asking for these things, and let's send them ahead.

So the trouble is all those features, and as I said, many of which are not even being used today, are in the spec. They're in the protocol. So you've got to build them in order for them to be used if you're going to say that you're HTTP/2. And everybody wants to be that now. And they are, it turns out, extremely tricky to implement. Netflix took a look at a number of actual real-world implementations of HTTP/2 deployed in the field and found eight different ways to completely clog up the pipes and bring even a burly web server to its knees with just a few carefully chosen packets.

What Netflix wrote was, in their overview: "Netflix has discovered several resource exhaustion vectors affecting a variety of third-party HTTP/2 implementations. These attack vectors can be used to launch DoS" - you know, now, this is not the flooding-style denial of service. This is a resource depletion. Remember back in the early days you could just send a server sequence, SYN, S-Y-N, packets, and it would start setting up connections which you never followed through on and exhaust its ability to create any more connections so that normal visitors coming were unable to get a connection. It would just return a "server busy" or just a spinning wheel rather than giving you a connection. Not because you were flooding it with traffic, but just you depleted the resources at the server end. This is like that.

So they said: "These attack vectors can be used to launch DoS attacks against servers that support HTTP/2 communication. Netflix worked with Google and CERT to coordinate disclosure to the Internet community. Today, a number of vendors have announced patches to correct this suboptimal behavior." And, by the way, there were a bunch of those in last Tuesday's Patch Tuesday from Microsoft. "While we haven't detected these vulnerabilities in our open source packages, we are issuing this security advisory to document our findings and to further assist the Internet security community in remediating these issues."

So under impact they said: "There are three broad areas of information security: confidentiality, so that's information cannot be read by unauthorized people; integrity, information can't be changed by unauthorized people; and availability, information and systems are available when you want them. All of the changes announced today," they

wrote, "are in the availability category. These HTTP/2 vulnerabilities do not allow an attacker to leak or modify information. Rather, they allow a small number of low-bandwidth malicious sessions to prevent connection participants from doing additional work. These attacks are likely to exhaust resources such that other connections or processes on the same machine may also be impacted or crash.

"HTTP/2, which was defined in RFCs 7540 and 7541, represents a significant change," they wrote, "from HTTP/1.1. There are several new capabilities, including header compression and multiplexing of data from multiple streams, which make this attractive to the user community. To support these new features, HTTP/2 has grown to encompass some of the complexity of a Layer 3 transport protocol," meaning the lower level protocol where there's a lot more going on than just get a connection, ask for something, get the results, and disconnect.

So they said: "Data is now carried in binary frames. There are both per-connection and per-stream windows that define how much data can be sent. There are several ICMP-like control messages - ping, reset, and settings frames, for example - which operate at the HTTP/2 connection layer. And there's a fairly robust concept of stream prioritization."

So what did they find? The description of the eight CVEs will give us enough of a feel for it. They wrote: "Many of the attack vectors we found, and which were fixed today, are variants on a theme. A malicious client asks the server to do something which generates a response, but the client refuses to read the response. This exercises the server's queue management code. Depending on how the server handles its queues, the client can force it to consume excess memory and CPU while processing its requests."

So there are eight CVEs, and I'll sort of run through their names and a brief description. So there's 9511, the Data Dribble. They wrote: "The attacker requests a large amount of data from a specified resource over multiple streams. They manipulate the window size and the stream priority to force the server to queue the data in one-byte chunks. Depending on how efficiently this data is queued, this can consume excess CPU, memory, or both, potentially leading to a denial of service."

Then we have the Ping Flood. "The attacker sends continual pings to an HTTP/2 peer, causing the peer to build an internal queue of responses. Depending on how efficiently this data is queued, this can consume excess CPU, memory, or both, potentially leading to a denial of service."

Then we have the Resource Loop. "The attacker creates multiple request streams and continually shuffles the priority of the streams in such a way that causes substantial churn to the priority tree." Once again, consumes blah blah blah.

Reset Flood: "The attacker opens a number of streams and sends an invalid request over each stream that should solicit a stream of reset stream frames from the peer. Depending on how the peer queues the reset stream frames, this can consume excess" blah blah blah.

Then we have the Settings Flood. "The attacker sends a stream of settings frames to the peer. Since the RFC requires that the peer reply with one acknowledgement per settings frame, an empty settings frame is almost equivalent in behavior to a ping. Depending upon how efficiently," blah blah blah, blah, blah, blah, blah, blah. Another denial of service.

The zero-length headers leak: "The attacker sends a stream of headers with a zero-length header name and zero-length header value, optionally Huffman-encoded into a one-byte or zero-length headers. Some implementations allocate memory for these headers and keep the allocation alive until the session dies. This can consume excess

memory, potentially leading to a denial of service." Then we have the Internal Data Buffering. And similarly. And the Empty Frames Flood and so forth.

So anyway, Netflix exercised existing implementations, found many of them wanting, and set up, got eight CVEs, coordinated the disclosure, responsibly disclosed these problems to the various vendors of HTTP/2 supporting servers, Microsoft among them. Microsoft fixed it last Tuesday. Presumably the other guys have or will, and we now have an improvement as a consequence in the actual implementation of HTTP/2. I mean, it is an incredibly complex protocol. But once the implementations get implemented and have been pounded on by researchers like this guy at Netflix, we'll have the potential for much better utilization of the bandwidth between browsers and clients, which we know everybody, especially Google, wants as they move toward web-based, browser-based applications.

Leo: Yeah. Hey, can I put in a little plug for Friday's Triangulation? I think you'll be interested in this. We've got...

Steve: Yeah, please.

Leo: Yan Zhu is lined up. She's currently Chief Security Officer at Brave, which is the privacy browser I've talked about built on Chromium.

Steve: Right.

Leo: She helped build HTTPS Everywhere, SecureDrop, Let's Encrypt. Really interesting person. She's also a DJ. She just performed at DEF CON. And she's going to be our guest on Triangulation this Friday. I'll be there - bcrypt. You know bcrypt.

Steve: Yeah, yeah.

Leo: I should have probably said bcrypt. More people know her by her handle, bcrypt, than anything else. We'll be talking - she was a Technology Fellow at the EFF, as well.

Steve: Very cool.

Leo: Dropped out of high school. Got her bachelor's from MIT in physics. It's an interesting mix, isn't it.

Steve: Love it, love it.

Leo: Yeah. So really excited. Former maintainer of HTTPS Everywhere and a co-creator of Packet City. Really an amazing person. 11:30 a.m. Pacific, 2:30 Eastern time this coming Friday I'll be talking to bcrypt.

Steve: That's be a great Triangulation.

Leo: Yeah, yeah. She'll be fascinating. Sorry. I didn't mean to interrupt.

Steve: Oh, no, I'm glad you did. I'm sure our listeners will be interested. I definitely will be, too.

I got a tweet from a David Theese who said: "Traveling from Phoenix on Thursday to hear your SQRL talk. Looking forward to meeting you. Long-time enjoyer of Security Now!." And just as a reminder, two bits. I am speaking the 22nd, in two days, Thursday, to the Orange County chapter of OWASP. And I have the link to the Meetup.com event. And also remember, I mentioned it last week, but Leo, you were not there, well, you were elsewhere, GRC.com/calendar is the link to the three upcoming OWASP presentations, the first one in two days here in Orange County, Southern California. And then I've got the one in Dublin, Ireland, followed by Gothenburg, Sweden, in late September. And I know that Rasmus Vind, who did the SQRL implementation for XenForo, I think I'm going to get to meet him in Sweden. And I'm going to get to meet Jeff, who did the iOS implementation of SQRL. He's going to come to Dublin. So it's going to be great. I'm going to get to meet lots of people.

Leo: Anders, is it Gothenburg? Or Gothenburg? Gothenburg.

Steve: Okay. No.

Leo: So I hope you get it right. Gothenburg. Okay. Okay. You're going to Gothenburg.

Steve: I'm going to just...

Leo: Just wanted you to know.

Steve: I'm - yeah.

Leo: And by the way, you'll be talking about SQRL at our presentation in Boston, too, and that'll be another opportunity for people to see you.

Steve: Yeah, cool. Cool. Also last week I explained I wanted to make sure that our listeners knew that two weeks ago we did Steve's File Sync Journey. On Episode 734 - which, Leo, you and I will be pre-recording before my trip...

Leo: Oh, I'm good. This is a good idea. Because we want to make something a little bit evergreen.

Steve: Yes, exactly.

Leo: This will be great, yes.

Steve: Exactly. So Episode 734 will be Steve's File Sync Conclusions. I've been keeping up with Twitter. I've been keeping up with the mail. I, oh my goodness, I mean, it's - I now am an expert on Syncthing. I even set it up on a Raspberry Pi and used Rclone and Mount in order to get Syncthing to sync to Dropbox, just as an experiment, so that it wasn't just peer to peer, but it was also to the cloud. And I've been playing with Sync.com and everybody else's, I mean, just a whole bunch of things.

So, and I'm happy to say things are beginning to gel. I think I'm developing sort of a, I mean, all of our needs are different. And so there isn't one solution for everybody. But I think I'll be able to present an updated sort of state of the where we are with applications and services and how they can be knit together, in some cases in some interesting ways that probably haven't been before. So that's a few weeks away, Episode 734. And, you know, this is our birthday, Security Now!'s birthday. And this is also the second anniversary of my first date with Lorrie is approaching.

Leo: Aww.

Steve: So I went looking for...

Leo: Now you know it's serious, if you remember that. That's good. That's a good sign.

Steve: Well, we were wanting to celebrate the second anniversary of our first date. So I went looking for the exact date from our early email correspondence, and I found it. And right next to it was a note from a SpinRite user. And I thought, oh. So I read it, and it was one of those little heart warmers. So I just thought I would reshare. I did, I'm sure, two years ago.

This is from Yann Fitzmorris. The subject was "Another Success Story SpinRite Data Recovery." He wrote to "Dear Steve and GRC Team." He said: "I purchased SpinRite a few years ago and have been using it to keep my drives in good health. I personally have never had to use it for data recovery." Well, of course, because he's using it preemptively. He said: "However, a friend asked for my help this week because her laptop would no longer boot to the login screen. Her laptop contained the only copy of pictures and videos of the first two years of her daughter's life.

"It wasn't looking good for the patient. When I plugged the drive into an external dock, no OS would recognize the drive. I used my dedicated PC for SpinRite, plugging in the drive, and ran Level 2. Success. We plugged the drive into the dock and were able to recover all pictures and videos, over 70GB." Whoa. "Needless to say, my friend will now seriously consider a backup solution. And," he says, "I'm hoping she will buy her own copy of SpinRite to show her gratitude for this amazing product. Thanks again for all your hard work and research. Love the Security Now! podcast, as well." He says: "I've been a listener since 2015." And Yann, maybe you're still listening, and here it is, 2019. So anyway, thank you for the nice little walk down memory lane.

Leo: A stroke from the past. Oh, we don't have an ad here, so have your cup of coffee, but we will continue.

Steve: I just did.

Leo: We will continue on.

Steve: Just needed to take a sip. I was dry. So this paper, the following paper was included in the proceedings of the 28th USENIX Security Symposium last week in Santa Clara, California. The paper's full title is "The KNOB Is Broken: Exploiting Low Entropy in the Encryption Key Negotiation of Bluetooth BR/EDR." Now, Key Negotiation Of Bluetooth is K-N-O-B, thus KNOB. Three security researchers reported on their analysis of more than 14 different Bluetooth chips from different vendors. If anyone is interested in the 16-page PDF, I have the link in the show notes.

So first of all, Bluetooth BR/EDR, that's basic Bluetooth. That stands for Basic Rate/Enhanced Data Rate, the original Bluetooth protocol, sometimes referred to as "Bluetooth Classic." For example, to distinguish it from Bluetooth Low Energy or something. That's present in more than, I mean, who knows how many, a billion, more than a billion Bluetooth-enabled devices. So the abstract of their paper explains, sort of sets this up.

They said: "We present an attack on the Bluetooth key negotiation protocol of Bluetooth BR/EDR. The attack allows a third party, without knowledge of any secret material such as link and encryption keys, to make two or more victims agree on an encryption key with only one byte of entropy." Yes, eight bits. Not many. "Such low entropy enables the attacker to then easily brute force the negotiated encryption keys, decrypt the eavesdropped ciphertext, and inject valid encrypted messages in real-time. The attack is stealthy because the encryption key negotiation is transparent to the Bluetooth users. The attack is standards-compliant because all Bluetooth BR/EDR versions are required to support encryption keys" - now, this is the part that is just difficult to believe. All Bluetooth versions "are required to support encryption keys with entropy between one and 16 bytes and do not secure the key negotiation protocol. As a result, the attacker completely breaks Bluetooth security without being detected.

"We call our attack Key Negotiation Of Bluetooth (KNOB). The attack targets the firmware of the Bluetooth chip because the firmware, the Bluetooth controller, implements all the security features of Bluetooth BR/EDR. As a standards-compliant attack, it is expected to be effective on any firmware that allows the specification and on any device using a vulnerable firmware. We describe how to perform the KNOB attack, and we implement it. We evaluate our implementation on more than 14 Bluetooth chips from popular manufacturers such as Intel, Broadcom, Apple, and Qualcomm. Our results demonstrate that all tested devices are vulnerable to the KNOB attack. We discuss countermeasures to fix the Bluetooth specification and its implementation." And that's the beginning of their 16-page paper.

CERT did a kind of a friendly explainer for this that I thought I would share. CERT's vulnerability notice, they use the common Alice and Bob as parties wishing to communicate securely, and Charlie in the role of attacker. They said: "To establish an encrypted connection, two Bluetooth devices must pair with each other and establish a link key that is used to generate the encryption key. For example, assume that there are two controllers attempting to establish a connection, Alice and Bob. After authenticating the link key, Alice proposes that she and Bob use 16 bytes of entropy. This number N could be between one and 16 bytes. Bob can either accept this, reject this and abort the negotiation, or propose a smaller value. Bob may wish to propose a smaller N value because he does not support the larger number of bytes proposed by Alice. After proposing a smaller amount, Alice can accept it and request to activate link-layer encryption with Bob, which Bob can accept."

They write: "An attacker, Charlie, could force Alice and Bob to use a smaller N by intercepting Alice's proposal request to Bob and changing N. Charlie could lower N to as low as one byte, which Bob would subsequently accept since Bob supports one byte of entropy, and it is within the range of the compliant values. Charlie could then intercept Bob's acceptance message to Alice and change the entropy proposal to one byte, which Alice would likely accept because she may believe that Bob can only support a value N of one. Thus both Alice and Bob would accept N of one and inform the Bluetooth hosts that an encryption is active, without acknowledging or realizing that N is lower than either of them initially intended it to be."

Leo: My key is 43.

Steve: So what we have here, obviously, and we've encountered this many times through the 15, or the 14, the now-completed 14 years of this podcast, another classic cryptographic security downgrade attack. It's amazing that Bluetooth is this mature, and that we are only now noticing this oversight.

Under impact, CERT notes: "An unauthenticated adjacent attacker can force two Bluetooth devices to use as low as one byte of entropy. This would make it easy for an attacker to brute force as it reduces the total number of possible keys to try, and would give them the ability to decrypt all of the traffic between the devices during that subsequent session." And here's the really nutty part. The researchers note the following about halfway through their 16-page paper, because I read the whole thing, and I was just like, you're kidding me. Well, not surprising, but they did say this: "We do not see any reason to include the encryption key negotiation protocol in the specification of Bluetooth."

Leo: Yeah, why would anybody want that?

Steve: "From our experiments, presented in Section 5, we observe that, if two devices are not attacked, they always use it in the same way. A device always proposes 16 bytes of entropy, and the other always accepts it. Furthermore, the entropy reduction does not improve runtime performance because the size of the encryption key is fixed to 16 bytes, even when its entropy is reduced." In other words, this was just, again, over-engineering. Someone said, hey, let's add a pre-link negotiation encryption key negotiation to allow the end points to negotiate down the amount of entropy.

Now, I mean, okay, was this the NSA who was secretly participating in some committee? Because this is nuts. I mean, this is loony. Anyway, it's what we have. And so a bunch of Bluetooth stacks are now being busily updated. Essentially, the problem is this is not the software. This is the firmware because this is all being done by the Bluetooth controllers. So we're going to need the Bluetooth software to update the firmware in the Bluetooth controllers to no longer accept entropy low values.

I mean, really, since these guys have never found a single instance in their testing where 16 did not work, where 16 bytes of entropy wasn't enough, that's what everybody should use. The whole point was - and this is why they said we do not see any reason to include the encryption key negotiation protocol in the specification. That's the protocol that allows the negotiation down from 16 to one of the entropy. It should just not be there. Nobody has a problem with 256 bits. So that's what it should have always been. Crazy.

And of course our long-time listeners will recall that I have several times observed that there is a large, though brief, period of inherent vulnerability during Bluetooth pairing.

You have two unauthenticated devices, that is, they don't know who each other is. They're just near each other. And they're saying "go" at the same time. So you have two unauthenticated devices, hoping to perform a secure negotiation. It's simply not possible to do that securely without some covert out-of-band channel. It's just not.

So I recommended, and our listeners may recall, that if someone really needed Bluetooth security, that they should stand out in the middle of a completely deserted parking lot to perform the pairing and hope that no one is aiming any high-gain antennas at them. Because that's the only way you can do this. I mean, maybe, if you really wanted to, just bring a large roll of tinfoil, wrap yourself in tinfoil, and then pair your devices under your tinfoil cone of silence, and you'd be good to go. But otherwise, I mean, the problem is it's super convenient to pair, you know, to make a device discoverable, and the other device sees it. But it's inherently an unauthenticated pairing process.

And there isn't any way to make it absolutely secure unless, as I said, you use some sort of out-of-band channel, you know, where for example one device has a screen and displays a long PIN, where you key it into the other device's keyboard. There you're out of band, not just relying on radio. But nobody wants to do that. You just want to say, oh, look, magic. We're paired. It's like, yes. To whom? No way to know for sure.

Leo: To whom are we paired? Somebody you should look at Bluetooth LE, which is crazy. I was driving down - I guess I was - what do you call Bluetooth wardriving? Bluedriving? I was bicycling through a neighborhood. I was trying to pair my phone to my helmet, my Bluetooth helmet. And all of these - I'm seeing, like, speaker after speaker. And not just speakers, I mean, all kinds of Bluetooth devices popping up here. And I don't think LE does any authentication; does it? It just goes, yeah, yeah, fine, good. We'll make this easy. Let's make this easy on both of us.

Steve: Yeah. There has to be a pairing event. So you do typically need to make both devices, like prime them to connect to each other.

Leo: These devices were broadcasting their Bluetooth identity, and I think most do because I think manufacturers say we don't want this whole pairing key thing. Let's just talk to each other. So there's a lot of new devices, Bluetooth LE devices that just talk to your phone. Like there's nothing, you don't do anything.

Steve: Wow.

Leo: I think that's crazy.

Steve: That is crazy.

Leo: Yeah. I would love to know more about that.

Steve: Especially when your house alarm security system is using it.

Leo: Oh, you wouldn't believe all the things I saw. Refrigerators, I mean, all sorts of stuff. Well, you know, I mean, this is that age-old tension between convenience and security.

Steve: Yup.

Leo: And the Bluetooth folks just want to make it easier and easier and easier. I think it has to do with beaconing. I think there was this push for a while to have these Bluetooth beacons. And it turns out every Apple device is a beacon. They just never told anybody.

Steve, we do this weekly. And you're still not jaded, 14 years...

Steve: For 14 years going. Here we are, the beginning of number 15 and going strong.

Leo: Happy birthday. I don't know, I have to look up what the 14th anniversary gift is. It's probably not anything nice. It's probably paper or something. But happy 14th anniversary.

Steve: Yeah, this actually - I did the TechTalk column for eight years in InfoWorld.

Leo: So this is the longest thing you've ever done.

Steve: Yeah. No relationship.

Leo: Including your marriage.

Steve: No relationship has lasted this long. Well, Leo, it's you and me, baby.

Leo: It's you and me. Okay. I'm just going to say the traditional 14th anniversary gift would be ivory. Now I know why the chat room kept talking about ivory. Gold jewelry or an opal, agate, or bloodstone.

Steve: Oh, Leo, you really shouldn't.

Leo: Gold is accepted, I'm sure. A Krugerrand or two would be fine. You'll find Steve at his website, that's GRC.com, the Gibson Research Corporation. That's where ShieldsUP! lives. That's where SQRL lives. That's where DCOMbobulator and Shoot the Messenger and all the great freebies Steve has been giving away over the decades. You'll also find his bread and butter, SpinRite, the world's best hard drive recovery and maintenance utility. Hie thee hither and get thou a copy of the beautiful SpinRite. You need it if you've got a hard drive. We will also have the show there. We're going to edit it, send it over to Steve. He puts 16Kb - do we send you 16Kb? Or do you down sample?

Steve: I down sample.

Leo: So he has a little script he's doing, taking the 64Kb, which he also has for an audio version, and puts it...

Steve: And then I put it up for Elaine because she likes to have the smaller one just to conserve her own bandwidth.

Leo: Oh, yeah, because she lives in the middle of nowhere. And so she types it in. Which is great because we sound like Thomas Edison in the first recording. So poor Elaine's going, "Hello, Elaine. We are talking to you through our microphones." Sorry, Elaine. She had to write that. The funny thing is, everything we say she has to write. That's kind of mean of me. I'm sorry, Elaine. GRC.com. Handwritten transcripts by a professional, 16Kb audio, 64Kb audio, all that great stuff. SpinRite.

You want video of the show or higher quality audio, we have it. Actually, we don't have higher quality. We have the same quality audio. But that's at our website, TWiT.tv/sn. That's the page. All the shows, all 14 years are there, 728 episodes. I'll give you a little tip. You can go to any episode by just typing TWiT.tv/sn and the episode number. So the very first episode is TWiT.tv/sn1, if you want to hear what we sounded like when Steve's moustache was still black.

What else? You can subscribe. Actually, I really encourage people to subscribe. We love it if you subscribe. Just get your favorite podcast client and search for Security Now!, and all of the shows will be yours. Thank you, Steve.

Steve: My friend, a pleasure. I will see you next week for 729.

Leo: Geez.

Steve: As we cruise further into Year 15.

Leo: We're entering our adolescence. Explains a lot. Thanks, Steve. We'll see you next time.

Steve: Bye.