Big Routing Tables

Description: After catching up with the week's more interesting security tidbits, Steve and Leo dig into last week's widespread Internet outage to discover that the Internet is reaching another important "limit" that's going to require some attention: The routing tables are growing past their maximum default size! Whoops!!

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-469.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-469-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson's here. He's going to explain the Internet outage that happened last week. It turns out it was just an accident, but it could be a portent of things to come. That and all the security news, next with Security Now!.

Leo Laporte: It's time for Security Now!, the show that protects you and your loved ones online, your privacy and all that stuff. Here he is, our Protector in Chief, Mr. Steven "Tiberius" Gibson, the man in charge at the GRC.com. He is the creator of SpinRite and been doing this show for a few years.

Steve Gibson: Yes, today is our ninth anniversary, and thus the beginning of our 10th year.

Leo: So this is actually an anniversary.

Steve: It actually is. It was 08-19-2005...

Leo: Wow.

Steve: ...that you and I sat down over Skype and did our first, #1, Security Now! #1.

Leo: Very awesome.

Steve: Yeah, yeah. And of course you had - and this was the second podcast. You had TWiT, the big flagship Sunday show. And you and I had talked about it months before when we were in Toronto. And I guess then we were doing them. Remember I would, like, bring up the Heil mics, and we'd sort of do an ad hoc deal, both in Toronto and then in Vancouver.

Leo: Yeah. I had a Mark of the Unicorn (MOTU) mixing board, little portable Firewire mixer. And we'd hook up the mics. We even did one in a caf, I think. I think.

Steve: I'd rustle the papers. I had stuff printed out, and we'd, you know, yeah, those were the days.

Leo: Ah, those were the days.

Steve: Ah, yes.

Leo: Well, I'm so glad we're still doing it. And of course the subject has exploded. Your show has actually grown faster than any of our other shows over the 10 years.

Steve: Well, and it's interesting, too, because I'm hearing you and Mike and, like, other of your podcasts referring to security issues. And I realized, I mean, it is, it's a topic that really has legs. It's, I mean, not only just for focused special interests, but, I mean, just in general, it's something that affects people.

Leo: More than ever now, frankly.

Steve: Yeah, yeah.

Leo: I'm sure we'll be talking about a few security topics today.

Steve: Well, so, yeah, today's topic is big routing tables, which sort of pushed itself to the forefront because the reason LastPass had problems that we discussed last Tuesday, we were talking about the LastPass outage, it turns out it was - and what we knew at the time was one of their datacenters went down. Well, it turns out that eBay and Twitter and a number of other major sites were having intermittent problems. And we now know - we know generically why, but we also know specifically what ISP pushed the wrong button that actually caused a major problem throughout the entire Internet, and why.

So I thought it's just a - it's a perfect opportunity to do sort of one of our fundamental technology podcasts, which are always popular. We've never explicitly talked about border gateway protocol and big iron routers. We talk a lot about residential routers, the little blue plastic boxes that we all have. And we've talked in the past about routing on the Internet. So we'll reestablish some of that background.

Then I want to talk about the consequence of the way the Internet has evolved, which

was not foreseen by the brilliant developers. Only what wasn't foreseen was that it was going to be this popular. I mean, they made some assumptions that were brilliant, which have really withstood the test of time. Some modifications have been made to sort of adapt to the way the 'Net has grown. But sort of the organic barnacles which have occurred have resulted in routing becoming substantially more complex than it really should have or might have. And then this also affects IPv6 in the future because - sort of in good and bad ways.

So we've got a great podcast to talk about sort of the fundamental technology of Internet routing and looking at, sort of retrospectively, what we've learned and what has happened over sort of history. And of course some interesting news. It turns out another presentation which will be happening at the USENIX conference occurring, I guess it's next week down in San Diego, is some researchers found another way of sneaking audio out of a room, even though there are no microphones on. We talked about the bag of potato chips or the tree shaking and showed that. Turns out there's another sneaky way to do it which users can't block until vendors fix that problem, and then they probably will be able to.

Some interesting legislation in Delaware. A site that is tracking information leakage. A welcome option in Java. And I wanted to talk a little bit about my newfound appreciation for the difficulty of writing really secure software, having emerged from the crypto portion of SQRL. And I understand and sort of pity people who really have to do it right because, boy, is our environment hostile to doing it right.

Leo: Interesting. Lots to come.

Steve: So a great, great podcast.

Leo: And I thought we'd change our album art. You know, we've been around 10 years. It's time to get a little different picture of you on the cover. So we thank Jeremy C. Waldecker for this. Something's wrong with this picture.

Steve: I described it as "deeply disturbing."

Leo: It's take on the picture we've talked about before of you and me as No. 1 and No. 2 on the bridge of "Star Trek: The Next Generation."

Steve: And I did post it on the Security Now! page at GRC. That's where Jeremy found it. Somebody else did the Photoshopping which put you and me in the bodies of Kirk and Riker. I mean, not Kirk.

Leo: Picard.

Steve: Picard and Riker on the Next Generation's Enterprise.

Leo: But now something's gone wrong. And Warf is saying in the background, "The transporter's acting very strange. Oh, no!" And there's been a mix-up in the factory.

Steve: Yes. It's what happens when the transporter malfunctions.

Leo: You'd look good with my hair.

Steve: I wish I had your hair.

Leo: You would. You would - that's a good look. Well, you could have it. You could have it.

Steve: I don't need it that much.

Leo: And I'm, you know, I'm relieved to know that if, should I lose a little of my hair, I'm still going to look okay.

Steve: Yeah, yeah. I mean, if you ask the barber for a No. 3 buzz...

Leo: I could get just that.

Steve: Then it looks deliberate as opposed to something where Mother Nature had her way.

Leo: Well, thank you, Jeremy. I love it. That's very, very funny.

Steve: For those audio listeners who didn't see what Leo just put up on the video podcast...

Leo: We tried to describe it, but there's nothing really you can.

Steve: Well, words really fail.

Leo: Yes.

Steve: What the transporter can do is hard to describe in some cases. So I'm not going to post it on the Security Now! page, but you can find it at the beginning of the video for this podcast, or in the show notes.

Leo: Yeah, it's in the show notes, yeah.

Steve: And the show notes are there, yeah.

Leo: Where do you put your show notes? You put them on GRC.com/securitynow? Is that...

Steve: Yup. They're on the same Security Now! page as the low-bandwidth audio and the links to your high-bandwidth audio and Elaine's transcripts. So everything is there.

Leo: And we can thank, by the way, Harry's for my lack of hair, apparently.

Steve: Or your lack of fuzz.

Leo: Lack of fuzz, yes. Did a little too good.

Steve: So these guys, security researchers who are doing a presentation at the USENIX conference coming up, named their hack the "Gyrophone" because it turns out that the gyroscopes which all of the state-of-the-art smartphones and smart devices have, the Android phones and iOS phones, iPhones, iPads and so forth, they're all - they have position-sensing technology, which is useful for, like, making the automatic switch between portrait and landscape. And of course gamers use these in order to allow you to sort of put position feedback into the game.

Um, and the technology involves some sort of a tiny little wafer floating in a capsule, which is probably sensed capacitively so that, as you move it, its inertia causes it to move relative to the base. And there's sensing technology which can sense that it has moved relative to the base, like the capacitance changes. It's very sensitive. What these guys postulate...

Leo: Is this the accelerometer they talk about on the phone?

Steve: Yeah.

Leo: Okay.

Steve: Yeah, yeah. And it actually isn't a gyroscope. That's sort of a misnomer, but it sort of has gyroscopic feel.

Leo: It's probably, I would think, a MEM, a MEMS - microelectromechanical machine.

Steve: Yes, exactly. It's a form of that. So they wondered whether there was and to what degree there was information leakage from that. Now, we talked about, back in 2011, so three years ago, talked about how some Georgia Tech researchers were able to decode the typing on a keyboard, you'll remember, Leo, from a phone placed on the desk not far away from the keyboard. And they could, like, recognize who was typing. And in some cases, if it was trained, they were able to do keystroke recognition because they were able to pick up the vibrations of the keystrokes.

So it turns out that some degree of speech is recoverable. The Android OS samples its - we'll call it a gyroscope because that's what it's generally known as, although it is in fact not gyroscopic, there's nothing spinning inside - the inertial sensors. Android samples it at 200Hz, whereas iOS samples theirs at 100Hz. And that ends up being a crucial difference. I wonder about the fact that they happen to be a factor of two, and the fact that Android came along later, and someone at Google didn't think, oh, well, we're going to sample ours twice as fast as iOS, just so that we have more responsive gaming because, arguably, that could make a difference just in terms of the smoothness of the feel. Although 200Hz is still easily fast enough for gaming, and I doubt that there's any huge difference between a hundred.

But in terms of audio recovery, what they found was, on Android devices - where, by the way, there is no user control over software access or even website access to the gyroscope. It turns out that Chrome and Safari on Android, the browsers themselves limit the JavaScript's access, thus a website's, a web server's or website that you're visiting's access to 20Hz. So there's no danger there. But Firefox for Android lets websites access the full 200Hz sample rate of an Android phone's gyroscope.

So these guys didn't spend a lot of time worrying about or working on recognition. They're not speech recognition experts. They make that clear in their paper, that they just - they only invested enough time to demonstrate that maybe Android should drop the sampling rate to 100Hz, or somehow control it more, because they were able, again, with their limited work, to discern the spoken digits, zero through nine, with an accuracy of about 65%. And they just sort of chose that because that's a credit card number being read is digits zero through nine. They could determine the speaker's gender with about 85% accuracy. And in a room with five different speakers, they could determine who was speaking with about 65% accuracy.

So that's impressive when you think of, like, only getting a readout at 200 cycles per second. I mean, that's way below the, like, normal 8KHz rate for audio that we would consider generally intelligible, or even 4K or 2K. I mean, this is a tenth of 2K. So remember we were talking about sampling relative to wagon wheels spinning last week. So you're getting very, very infrequent samples relative to what we can normally consider audio. So it makes sense, I mean, you're not just going to play that on a speaker and hear someone's voice. You're going to process it with software and do a lot of filling in of the blanks and essentially train it to recognize, like, what does the number nine look like at 200Hz? It's not going to sound like anything, but what's it going to look like?

And so this is the kind of thing, actually it struck me that this is - the sort of things that people seem to be doing now, maybe it's only post-Snowden that I'm thinking in terms of, like, what somebody with a budget could do. But this is the kind of stuff the NSA could do where they're probably thinking, darn, there goes another one of our channels. The security guys have found something else that we were having fun using that nobody would ever suspect, the inertial sensors, the vibration sensors, essentially, in our smartphones. So they actually are leaking information. It's not super troublesome. I wouldn't be surprised. There were Google people present. Wait, I'm using the past tense.

I'm sure I read them saying there were Google people present, but I didn't think that the conference was yet. Anyway, I know that one way or another Google knows about this, and so they may very well do some mitigation, just because they don't want to have any leakage of that sort.

And this was sort of interesting. I wanted to talk to you about this. Delaware just enacted some legislation that had been proposed by a nonprofit group. The Uniform Law Commission is a group that tries to sort of keep our federated states synchronized so that we're not wandering off too much in different directions. So they proposed some legislation, this Uniform Law Commission proposed some legislation with the awkward acronym, the UFADAA, which is the Uniform Fiduciary Access to Digital Assets Act. And what this does, Delaware put a bill together which the House passed - it was Delaware House Bill HB 345 - which the governor signed into law, which classifies digital assets in the same fashion as physical assets after a person dies, meaning that your survivors get access to all your social media accounts and other digital accounts.

The way the law reads, it says: "A fiduciary with authority over digital assets or digital accounts of an account holder under this chapter shall have the same access as the account holder, and is deemed to (i) have the lawful consent of the account holder and (ii) be an authorized user under all applicable state and federal law and regulations and any end user license agreement." So essentially, if you pass away or are incapacitated, in the same process of probate where your other assets are inherited by your heirs, under this legislation so would your digital assets be, which I don't know that I want that.

Leo: Well, there you go. That's your fiduciary responsibility.

Steve: Currently, it's only Delaware. And there's discussion about whether it's where the person is, or where the company is, because there was talk about California being important because we have so much happening in Silicon Valley. But, and I'm hoping that a will could override this, where a will could say, for example, right now this is Delaware only, but this seems to be the direction we're going in. So do we want our heirs to have all of our usernames and passwords?

Leo: The real issue, I think, is sometimes an heir will go to Facebook, and Facebook will say - well, actually Facebook has a process now. But originally Facebook would say, well, I'm sorry, you're not the person. I don't care if you're their spouse or you're their executor. We're not - you know where it's happened with? Gmail. In fact, it's happened a couple times with email systems for soldiers who have lost their lives.

Steve: Yes. There was a famous Yahoo! case...

Leo: Right.

Steve: ...where Yahoo! refused to turn off...

Leo: And his dad wanted to access his final emails, and Yahoo! said you're not him.

Steve: Right.

Leo: So I think it's probably more to address that kind of situation; right?

Steve: Right. So typically, when a person dies, access to a digital service officially dies with them. And with this law, your digital life would become an asset which your heirs could inherit. And so as I was saying, I'm hoping that you can override this with a will that says I don't want my heirs to have access to my accounts. I mean, that just sort of seems creepy to me. It's not a problem for me since I plan to live a long time, and I don't have any kids. But still - or my family, it's like, my stuff is not anyone else's business. I don't want someone posting as me or impersonating me.

Leo: We've talked about this, in fact we just last week talked about it on This Week in Google. And Google has an account activity monitor. They don't mention death.

Steve: Yes, I did hear you guys talking about that, yeah.

Leo: But they say there's a certain - you can set the period of time, if your account has not been accessed. And one of the behaviors is we'll delete everything, so even if your heirs do have access to it.

Steve: Yeah.

Leo: Does the law specify that I have to give somebody my passwords? Because if they don't have my passwords, then they have to go to Google; right?

Steve: No. No. And in fact many Terms of Service specifically exclude that. In Facebook's Terms of Service they said you will not - in their Terms of Service you are agreeing you will not share your password, or in the case of developers your secret key, let anyone else access your account, or do anything else that might jeopardize the security of your account. "You will not transfer your account, including any page or application you administer, to anyone without first getting our written permission." So Facebook is stating very clearly that this is for your personal and private use and not to be shared.

Leo: Although they do have a process, if you die. Your next of kin will then notify Facebook you died, and they make a memorial, they do something called "memorialize" your page. Did you know that?

Steve: Which, like, freezes it? No.

Leo: No, it allows people to comment. But I can't remember what the rules are. But it's like this person has passed on. This is a memorial now to that person.

Steve: Well, so we're figuring this out, I mean, as our lives become digital, and that digital existence outlives us.

Leo: It's happening. Now, so far this is Delaware only, and I don't know if this is...

Steve: Right. Although from the background that I read, it looked like they're just first. Delaware tends to be where a lot of companies incorporate. My previous company, Gibson Labs, was a Delaware...

Leo: Yeah, TWiT's a Delaware corporation.

Steve: ...an S-Corp because they tend to be corporate friendly, and I just didn't bother with Gibson Research.

Leo: So here's the Facebook page, what happens when a person's account is memorialized. No one can log into that account again.

Steve: Okay, good. So it's locked.

Leo: It's locked. It can't be modified in any way. This includes adding or removing friends, modifying photos, or deleting any preexisting content. Depending on the privacy settings, friends can share memories on the memorialized timeline. Anyone - I don't know what the purpose of this is, but anyone can send private messages to the deceased person.

Steve: Ooh.

Leo: That's odd.

Steve: Where do they go? Where do they go?

Leo: What happens to them? Content the deceased person shared remains on Facebook and is visible to the audience it was shared with. It's just kind of frozen in amber. But memorialized timelines don't appear in public spaces, such as suggestions for people you may know, or birthday reminders. Seems like a good idea. So I think Facebook has really been on the forefront of what do we do because it's probably happened quite a bit already.

Steve: Yeah.

Leo: Yeah. You know, it has to be a federal law. A state-by-state solution is not

going to ever work. Until it's a...

Steve: Right. And I think, you know, this tends to be the way things happen is that states experiment with them and play with them, and then at some point someone decides, yeah, you know, that seems to work. Let's have this go nationwide.

Leo: Yeah.

Steve: So there's an interesting site that I thought would interest our users. Mike talked about it this morning on TNT. He just mentioned it briefly. I wanted to give it a little more attention because it's sort of our focus. And it's called "HTTP Shaming." It's at httpshaming.tumblr.com.

Leo: We talked about it on TWiT.

Steve: Oh. And so he just put it up, that is, the person who created it. And individuals are able to submit sites which they feel are not providing the security that they wish they had. I'm impressed by it because, although it's a little knee-jerky, over-the-top, on the other hand, the information is quality. There are packet traces from Wireshark showing the URLs and the data in the clear and so forth.

So, for example, for Verizon Wireless - I picked some sort of major high-profile sites. Verizon Wireless, this site mentions that they use a nonsecured homepage which prompts for login with a form. And we know why that's not safe. Even though the form might be safe, that is, the submission might be an HTTPS URL, if the page hosting the form is not secure, you don't know if the contents hasn't been changed because it isn't over a secure tunnel. It came to you in the clear. And anybody in a Starbucks who is able to intercept that could change it so that it looks like the Verizon Wireless page, but you're logging in to them, or they're intercepting your credentials. Or all they had to do is take the "S" off the HTTP in the form submission.

Leo: Wow.

Steve: What?

Leo: Oh, no, I'm just saying wow.

Steve: Yeah.

Leo: You know, there is some good news here because they called out 1Password, for example, and they immediately fixed it.

Steve: Yes. And in fact TripIt was one that caught my attention because TripIt is a

popular trip-planning site. And it turns out that it doesn't use security at all. So I was listening to you talk, I have been listening to you talk about how arguably TWiT really has no need to be over HTTPS, which I agree. For people who are just browsing for content.

Leo: Well, I want to ask you about that. You saw all the tweets from people showing this man-in-the-middle attack using kitty videos. Right?

Steve: Yeah, yeah.

Leo: Good. I want you to reassure me. We'll address that later. Okay.

Steve: Yeah. So but in the case of TripIt, they're never using security, yet it's an interactive server which is sending your potentially sensitive travel details, calendars and so forth, all in the clear. That is, everything you transact with them is unencrypted, and maybe you care about that. I mean, it's not just them being a static site, delivering the same page over and over and over to different people, where you could argue there's nothing there that anybody else couldn't get if they went there. Essentially, any time a website is providing content for you, customized in some way for you, I think the argument could be made that should be secure because that's something you might not want other people to have access to. My point being that TripIt immediately responded that they're going to fix this.

Leo: Good.

Steve: So, and Scribd is here, Ubiquiti, DirecTV, KeePass, ASUS. So there are some rather high-profile sites that are being listed on this. They're getting attention. They're generating feedback from users. And unfortunately, this is what it takes. Sites won't move unless their users say, look, you don't fix this, you're losing me. And I know from our podcast listeners there are a lot of people who feel that way about the sites they rely upon. So although this is a simple idea, HTTP shaming, it's looking like it's going to be socially effective.

Leo: Yeah. So I don't - I should look and see if it's in your notes. Did you want to talk about this issue of whether I should encrypt?

Steve: Yeah. I'm still unclear why it's important for you.

Leo: So a bunch of people tweeted me and you saying, "Now how do you feel, Leo?" And they quoted a security researcher who was showing how a YouTube video - now, it requires a man-in-the-middle attack, but how a man-in-the-middle attack could be used to inject malware into a YouTube kitty cat video. And the point being that, if there's a man-in-the-middle attack going on, my traffic could be modified on its way to hack somebody's computer, and that would be eliminated by an HTTPS connection.

Steve: That's true, of course, for any non-HTTPS session between a user and their browser. So I guess the argument would be we should ban all nonsecure traffic. Well, that's not going to happen. I think a really reasonable compromise is to understand that forms should be delivered securely in addition to their submissions obviously being secure. And as I said before, any web app sort of session, where you're inherently transacting, whether it's reservations or trip planning or social anything, that's about you, where you're receiving customized pages, I think that ought to be underneath the cone of silence. If you're looking at static pages, where everyone who goes gets the same one, yes, a man-in-the-middle attack could alter that page and inject something into your browser, which has always been true. And the only way to resolve it - so it's not more true about you than it is about anybody else.

Leo: Would global HTTPS eliminate this problem?

Steve: Yeah, actually that's the only thing that will because, I mean, if you're going to go all the way, then you can never allow a non-HTTPS connection, or that could be a way for them to get a shoe-in. That is, if you ever allow a non-secure connection, and if that were modified to take the S's out of all of the HTTPs, then your user would just assume your site was not secured, when in fact you intended it to be, but you were waiting for their browser to send you HTTPS queries, and it didn't because somebody got in first and removed all the S's from the page. Like a little bit of evil script could do that easily.

So in fact GRC did this a couple years ago, and Adam Langley, of all people, was nice enough to put my domain in Chrome so Chrome knows never go to GRC except securely. And of course the onus now is make sure I always have a valid security certificate. And again, that's the tradeoff. It's very easy for people to say, oh, everybody should be secure. It's like, try that. Try it for a while. You've got to be on your game. And as I said, the certificates are not free. If you want to get good ones, they're more expensive. And then you really need to make sure you don't lapse and let them expire or nobody could get to your site. So we're clearly moving in that direction. And I think it makes more sense to say, "Secure things where it makes sense to secure them."

Leo: Although that's the point, that if a man-in-the-middle attack could be used to propagate malware from a site that's not secure, then every site needs to be secured, I guess.

Steve: That's my point because it's like, there's nothing different about TWiT from anybody else.

Leo: Well, you watch video on the site, so somebody could inject something bad in it, I guess.

Steve: Okay, yeah.

Leo: But couldn't you also, if you've got a man-in-the-middle attack running, couldn't you just replace my certificate anyway? Intercept it and then...

Steve: No, because assuming that the person's browser hasn't been compromised, there's no way for some third party to get a TWiT.tv certificate. They'd have to convince a CA to give them one, and CAs are, I mean, their whole job is not to give bad guys certificates for other people.

Leo: And maybe we should get an HTTPS certificate. What the hell.

Steve: Yeah, I mean, now that Google's made the switch, we're seeing more and more companies, I mean, just during the last couple years Facebook has gone, Twitter has gone, Google has gone. GRC, I was in no big hurry because I was just, I mean, I have very little, only the purchasing of SpinRite actually involves any interaction with the user. Otherwise it's just pretty much static pages. But finally it was sort of like, well, okay, we're about security, so I should be secure.

Leo: Yeah. And so should we. And a number of people sent me notes saying, well, you can get a cert for five bucks or 10 bucks or whatever a year. I mean, there are cheap certs out there.

Steve: Yes, yes. Oh, and it was - I referred to - it was funny. I referred to STARTTLS, and I thought, wait a minute, it's not STARTTLS because that's an email protocol for bringing up a TLS connection over POP or IMAP or so forth. It's StartSSL. They're a...

Leo: Right.

Steve: StartSSL will give you, I think it's a one-year certificate. They just set the expiration to one year because they can and because it's free. Though it's not universally accepted in all browsers. And I did hear that some browser was removing their - like Mozilla or somebody was going to stop honoring StartSSL certs just because they were potentially flaky.

Leo: Well, because they have a very ugly website. I think that right there is one reason. That is an ugly ass website. Jeez. All right.

Steve: I'm not commenting because, you know...

Leo: Yeah, no, you should stay out of this one. Hey, yours is functional, man.

Steve: It's very clean, yes, functional. Okay. So there was great news - our friend Simon Zerafa tweeted to make sure that I saw this and forwarded me the link - the news that, with the latest update of Java, something new got added to the control panel. Down there at the bottom, second from the last line under Miscellaneous, was something I was so glad to see. Under Advanced on the Java Control Panel, under Miscellaneous, there is now an option, not checked by default, of course, but it's there, which reads: "Suppress sponsor offers when installing or updating Java." Yay.

Leo: OMG.

Steve: Yes. I know.

Leo: But it will only be effective for future installs.

Steve: Yes. And it's only going to help the cognoscenti among us because it's not checked, obviously, by default. So for everyone listening, for those of you who need Java - I need it. I don't have it in my browsers, so my browsers won't run it. But I use the Eclipse platform and a number of other things that are Java-based. So I've got it in my machine. And I'm always getting that Ask Toolbar offer that's so annoying. And once I forgot to turn it off. I just was in a hurry, and I click, click, click, it's like, ooooohhh. So go into the Java Control Panel and just enable "Suppress sponsor offers when installing or updating Java." And from then on the wizard skips over that page, or defaults the checkboxes to off rather than on. So it's no longer incumbent upon you to make sure you never install that by mistake. And again, so I salute Oracle. Thank you for that.

Leo: Well, it's about the least they could do. Literally.

Steve: But at least it's there. Microsoft had a little booboo. They had a problem with one of last Tuesday's updates, which was causing restart problems for people. It's some tangle involving fonts being installed in nonstandard directories. It's not really clear what the problem is. But they have withdrawn it from their site. And under their - I'm not sure how you get to the Knowledge Base article if your computer won't boot. But if you are able to, maybe you do your previous configuration option, emergency restart cycle or something. But they explained: "Customers who are experiencing difficulties restarting their systems after the" - I love this, the way Microsoft phrases things. "If you've died recently...." Well, no, "restarting their systems after the installation of security update 29827..."

Leo: Oh.

Steve: It's a star date. Star date "2982791 should refer to the Known Issues section of Microsoft Knowledge Base Article," and same number, 2982791, where it sort of explains that there seems to be a problem with fonts, and they're not really sure what the problem is, but they've identified the sensitivity with four other updates. So you're supposed to go through and uninstall those. And it's a mess.

Leo: It's not merely a security flaw, though; right? It crashes the computer and stuff.

Steve: Yeah. You can't get into Windows. I mean, and that's, yeah, I mean, it...

Leo: Okay. Hmm. Minor detail.

Steve: It's a BSOD sort of nightmare, and not good. So Microsoft has withdrawn it, and I guess it'll probably emerge a month from last week for the September patch cycle.

I am done with the - I'm almost done with the UI stuff in SQRL. This, Leo, I have it right here, this is, as of a couple days ago, the page that SQRL prints when you press the "Print My Identity" button in the SQRL UI. There is, in the top of the page, is a QR code format of a SQRL identity. And then lower down, in case you are ever in a situation where you don't have a webcam, is a short sort of textual version that if, in an emergency, somebody had to type in, they could, in order to recover your identity.

So anyway, it's coming along nicely. And so all of the crypto work has been finished now for a few weeks, and people have been testing it. Ralf's Android client, which is sort of a pro forma client, available over in the Google Play store, and SQRL are able to - and my client are able to interchange identities. I won't get into a lot of the super details because a lot more of this will be automatic than it is right now. But it's beginning to come alive.

I posted in the newsgroup my newfound appreciation for how really difficult it is to write really secure code because - and I've been coding now for more than 40 years. And so I'm to the point where I sort of have the philosophy of coding. And when I find a bug, I spend some time asking myself how that happened because I'm interested in the process as much as the result. And what I appreciate is that the architecture of our computers is really lagging behind what we want from them from a security standpoint. There is some architecture enforced by the hardware, the notion of processes where a process runs within pages where the chip enforces the paging so that processes have - essentially what that means is they have isolated address spaces so that the address space in one process is disjoint from the address space of another.

Now, unfortunately, there are needs that we have for breaking that process isolation. For example, processes have to be able to talk to each other in order to get work done. Debuggers have to be able to reach into another process in order to halt it and single-step it and allow a programmer to examine what's in there. So that means the operating system, to support cross-process debugging and cross-process communications, has to in some way break that isolation. But at the lower level, just within the process itself, it is extremely difficult to keep anything secure from leaking.

So, for example, as I was writing SQRL, there's this concept of local variables, which are variables that are allocated on the local stack, which is very quick and easy to use. But they're sort of - they're sometimes called "automatic variables" or "local variables." They sort of disappear when you leave the procedure or the subroutine where you were using them because they were dynamically created just by moving the stack pointer down such that the region above the stack pointer is now sort of available as a scratch pad. And when you leave the subroutine, the stack pointers move way back up again, which sort of has the effect of just discarding those. But they're not erased. They're still there on the stack.

And so for my definition of what truly secure code is, I refused to ever leave any sensitive data on the stack when any of my subroutines exited, which sometimes got me in trouble because I received information on the stack when a subroutine was called. Then I had to decide whether I could erase it myself, so, like was I through with it? Or did the caller retain responsibility for erasing the sensitive data that it was passing to me? And, I mean, my point is that coding - and I've never had to, you know, nothing

that I've ever done has had this level of insistence from me of, like, rigor in security.

The server at GRC assumes, because I have really good control over all of it, I can make the assumption that, within the server, I have security, certainly within the web server process. And it's never been the case that that's been a problem. But a desktop where this SQRL client is operating is just - it's the Wild West still. I mean, it's still a scary environment. We were just talking about it, like evil kitty YouTube videos, the fact that those exist is disturbing. And so...

Leo: And you're surprised?

Steve: It's not that the kitties are evil, of course.

Leo: No.

Steve: It's just that malware can use that to ride into your system. So I'm determined that my client will be ruthlessly secure. Which means that, for the briefest period possible, things are decrypted. And anything that is sensitive is overwritten before it's released. And so I've, in several cases, I've gone back through my code to - and, like, I left breadcrumbs for myself saying, okay, remember, in every instance where I ever decrypt something, make sure that I zero it before I leave. Don't just stop using it. Wipe it.

And oh, my lord, I mean, I'm sure that isn't being done most of the time in this industry because nothing would get done. I mean, there's, like, no help for this. It is incumbent on the programmer to understand, carefully understand the scope of use of sensitive data and take responsibility for proactively zeroing it before they stop using it. And that's just a - that's a much higher barrier for coding. SQRL enforces that practice throughout all of the crypto work it does.

But, boy, I really came away with a much more deep appreciation for how difficult that is. And most people are not writing assembly language at the register level, managing this themselves the way I am. And my point is that most of the time you're using third-party libraries. So you really don't have control over what's being done outside of there. And I referred to something a couple weeks ago that I ran across, it was code - I don't remember now what it was. I think it might have been in OpenSSL. I saw something that they weren't doing, and it was like, ooh, boy, ouch. And I have verified, since I've assembled the library SQRL is using from their source, I have made sure that memzero is a typical call, that they are zeroing any buffers that contain sensitive data prior to then being done with it.

And the good news is the authors of those libraries knew what they were doing and understood the need to do that. But, wow, it's, I mean, I have, like I said, a deeper appreciation for how difficult this is. And, as a consequence, an understanding of the fact that just most code isn't going to be secure. I mean, not within such tight boundaries. For example, if you rely on the process to be secure, then it's like, okay, within the process you can be sloppy, if you assume that your process boundaries are - the security at the process boundary is being enforced reliably by the OS. I'm not making that decision, so I decided I didn't want code to be able to get injected into the SQRL client and catch it when it wasn't looking leaving sensitive information on the stack. So I never do. But, boy, it's a high bar. And it sure does, I mean, it's fun to do it, but I don't want to

do it again. I'm glad I'm through with that portion of it. It's just work.

When I'm going through the mailbag for our Q&As, I sometimes encounter, but I don't really have an opportunity to address, the question that people ask when they say, "What does SpinRite do?" And since we didn't have too much news this week, and I can easily fit our discussion of big routing tables into half an hour, I wanted to take a little bit of time for the sake of listeners who haven't been listening that long. Because invariably they say, I see email that says, "Hey, I've only been listening for a while, and I hear you sharing testimonials from people who report that SpinRite recovered data that they desperately needed to have recovered. But, okay, how?" Or "Why?" Or "All you ever do is share success stories. You never talk about what it is."

And so I've sort of had that comment from people. So I thought I'd spend a minute just to sort of explain, because it's an interesting fact, that I never designed SpinRite for data recovery. That wasn't what it was for.

Leo: That's why I always call it the "best hard drive maintenance utility."

Steve: Right. And it sort of had a maintenance mode. But even that was sort of a side effect. So the way SpinRite began was when - it was actually generated by my noticing that - there was an old-style, servo-based hard drive that actually a girlfriend of mine had all of her company's bookkeeping on, years of financial records, and it failed. And she was desperate to have this data back. And so I said, okay, I'll take a look at it. And this was one of those big, 5.25-inch, full-height, full-width, really heavy, servo-based drives. And I don't know what inspired me to do this. But I guess I was desperate. But I lifted one end of the drive up, and then it worked.

Leo: That's bizarre.

Steve: And I knew why, because there was...

Leo: All the bits were rolling down to one side.

Steve: They were loose. Exactly right, Leo. They needed to sort of shuffle around down there and reorganize themselves.

Leo: They were all sliding to the end.

Steve: There was something known as "tower drift," where a hard drive would have a servo platter which one of the heads would read. That told it where the tracks were. And then all the other heads essentially got their positioning information, just physically, because they were connected to the same mechanism. And what would happen, though, over time, just due to mechanical wear and tear, heating and cooling the drive up every morning and evening, is that sometimes the head that was furthest away would go off track. That is, it would stop being able to find the sector headers that were originally written when the drive was low-level formatted.

And so what happened when I lifted one end of the drive up about an inch is that put some gravitational bias into the side-to-side motion of the heads. It literally, it pulled the heads toward the center of the platters because gravity created a bias. And now we were able to find the sector headers.

Leo: Wow.

Steve: So what I realized was...

Leo: You knew that? I mean, you just kind of knew that in your head?

Steve: Yeah, well, I'm an engineer, so I understood that.

Leo: Wow.

Steve: So then what I knew I had to do was I had to refresh the low-level format of the drive and essentially migrate the sector headers back to their new position that they would have when the drive was level again. So I quickly wrote a program which read a track, re-low-level formatted that track, and then wrote it back. And the act of doing that - oh, that was while the drive was elevated, while it was able to find its sectors. And then I ran through that process. Then I lowered the drive about a quarter of the way and did it again, and lowered it another quarter of the way and did it again, and the final quarter of the way and did it again, and then horizontal and did it again.

Leo: Wow.

Steve: And so what that did was, with every quarter of an inch of lowering, the tracks were actually moving back out to where they would naturally be. But by low-level formatting the entire drive each time, I was pulling the headers with me. I was bringing them back into alignment. And it fixed the drive.

So it was not longer after that that I stumbled upon the idea - or I did something called SPINTEST and SPINTIME when I was writing the InfoWorld column. These were very short programs, a little bit of BASIC with the program itself in data statements of hex at the end. And so the program would just poke the data into memory and then jump to it. And that allowed me to deliver, in a printed page, it allowed me to deliver assembly code.

And SPINTEST and SPINTIME were simple algorithms which just read sector one over and over and over and checked to see how long it took. And by doing some math - and also doing, like, sector one and sector two, sector one and sector three and so forth. Then, doing some math, I could figure out what the interleave of the drive was. And what I and many hundreds of people who used these little pieces of freeware discovered was that the interleaves were wrong. The drives at that era, MFM drives, had 17 sectors per track. And the computer could not read sector one and sector two and sector three sequentially. It was only able to read a sector at a time, so it would read sector one. And then, after reading it, the controller would transfer it into RAM, and then the computer

would ask for sector two.

Well, the problem is sector two, if just even the first little tiny bit of sector two had started to go past the head, we had to wait for it to come all the way around again to get sector two. Then we would do that. And then, by the time we got to ask for sector three, it had already started to go past the head. So again we had to go all the way around.

So this problem was well understood. IBM, in the XT, they set the so-called "interleave" to six so that you'd have sector one, and then six physical sectors upstream would be logical sector two. So remember that, like, the physical sectors are the actual physical location around the track. But each one of those sectors has a header that declares "I am logical sector" one, two, three, four, and so forth. So if the interleave was 1:1, then that would mean that the physical and logical numbering was identical. But if it was, like, 1:2, that would mean that every other physical sector was logically sequential. So you'd have sector one, and then, I don't know, like sector six, and then sector two, and then sector seven, and then sector three, and then sector eight and so forth. So that would be with an interleave of 2:1.

XT shipped it with an interleave of 6:1, which meant that - essentially, that gave the computer ample time to read in sector one, digest it, and decide it wants sector two. And by that point, instead of sector two already just having passed by, it would still be out in front, coming toward the head. So that's much better than them having it just underneath the head or leaving. It would still be on its way in. So what that would mean is, with an interleave of 6:1, you could get every sixth sector in a single revolution, or that reading a 17-sector track took six revolutions because you were only getting every sixth one each time.

Well, the clones, the famous WD1003 controller that all of the clone computers, the IBM PC/XT clones used, the Western Digital controller had an interleave of 3:1. And I'm sure that some engineer at WD thought they were being really clever. Maybe they were running in an AT, like in a faster machine. I never understood why WD used 3:1 except that it made them twice as fast as IBM. That is, if you used their controller with an interleave of 3:1, it would take three revolutions of the drive to read the whole track, rather than six. Unless that particular clone that you put that Western Digital controller in couldn't handle 3:1, which was always the case.

It turns out those - all of the clones that everyone was using needed 4:1. And in fact 3:1 was the worst possible interleave because that meant that, when the computer asked for the next sector, it was standing on it. It wasn't just in front of it, that sector wasn't just in front of it, it was on it. So if the disk was interleaved at 3:1, it took 17 revolutions to read the entire 17-sector track because you were only getting one sector per revolution. It was awful. And the whole world was putting up with that. Nobody understood this. But I understood that that's what was going on. The only way to fix this after you're using your computer and you've got data - and remember, the only other backup we had was floppies. And so people used to, like, if you had to back up your 20MB or 30MB drive on diskettes, that was something you really put off.

Leo: Did we do that? Did we really?

Steve: Oh, yes. We did. Remember stacks of disks? You'd just put one in after another. And of course floppies were so unreliable that you had no real guarantee that your backup was going to be usable if you ever did need it. So the only recourse someone would have would be to back up their whole drive and then re-low-level format at the -

because the low-level formatting time is when you set the physical location of the sectors, and you could override with some commands. You did G=C800:5. C800 was the segment in memory; :5 was the offset of the starting address of WD's interactive sort of command line tool to do this.

And then you could give some commands, and it would low-level format. And you could say, I want an interleave of four. Which was fabulous, except you couldn't do it with data on your drive. So then you'd have to - so you'd have to backup the drive, pray that your floppy backup was good, re-low-level format the drive, and then put everything back. Unless you got a copy of SpinRite. Because what I realized from my experience low-level formatting that 5.25-inch full-size drive was there was a command that allowed you to low-level format a track, just one. And that meant that I could incrementally re-low-level format a whole hard drive by doing it one track at a time.

I would read the track, interleaved at whatever it was, then re-low-level format it at the proper interleave, and then put the data back, and then go to the next track. If people did that, they would - their computer would go from reading a track in 17 revolutions, this is all the clones in the industry, like virtually all the computers, took 17 revs to read a track. After running SpinRite, it was four. So that was a 425% improvement in speed. I mean, you could feel that. I mean, the computer just booted, like, whoa, holy - I mean, people were amazed.

The problem was that there were defects on these drives, and manufacturers were supposed to note the location of the defects at the time that the drive was low-level formatted. Many of them didn't. But more importantly, what I'm doing when I'm changing the low-level format is I'm physically relocating the sectors to different positions. But that means that, if there was a defective sector in a certain location, like sector five, that logical sector five that might be marked in the file system as "do not use," that becomes logical sector 10, for example, if the interleave changes. Meaning that the defects are going to appear in different sectors of the file system if I change the interleave.

So that was sobering because that meant, if I was going to do this, and our listeners now know, having just heard about what I've done in SQRL to make sure that the code is secure, I'm very methodical. I'm going to do it right. So this meant that I needed to perform defect analysis to make sure that it was safe to put what was a good sector's data in a different physical location that may not have been marked out by the manufacturer. And essentially the defects were moving because I was moving the sectors around.

But the other thing that happened is, if I was going to do a low-level format, we know that the one thing a low-level format does is wipe out what's on the track. I mean, it plows, it just plows a new data zone for the data to be laid down. The head is turned on, and it just - it wipes out what's there. Well, that meant it was also incumbent upon me, who insists on doing as perfect a job as humanly possible, to make sure I have read the data that is there. That is, to recover what is there no matter what, because once I low-level format, it's gone.

So what I did, way back at the beginning, like 23 years ago, when I wrote SpinRite v1, was I built in really robust data recovery technology. I mean, I did everything I could imagine. Back then, we often had stepper positioning, sort of dead-reckoning positioners, where it was just a stepper motor that operated in notches. And it had a rack-and-pinion that moved the head out to the proper location. But it was just that mechanical system that decided where the head came to rest. There was no servo back then. That was only on the really high-end drive, the one that everyone used, like the famous Seagate

ST225. That was a 20MB drive, and it just had - it had a steel tape wound around a round hub on the stepper motor, and that steel tape positioned the head.

So I needed to, in order to make sure that I got all the data that was there, if I had a problem reading the sector, I would do things like step out in one direction, then come back and try it again, and then step in the other direction a different distance and come back. And in fact I have a random number generator, there's always been one in SpinRite, to choose a random distance to go away in order to come in at a different arrival velocity and profile in order to get the head in a slightly different position, maybe to coerce the drive to give it to me one last time. I just need it one more time.

And so all of that is in there. And there's even technology where, if I absolutely finally am unable to get a sector to be read completely correctly, then there's a way for SpinRite to say, well, give me what you've got. Because it turns out that there was some benefit to the "give me the best you can do." For example, you might have a sector that was in a file system where there's only a few entries at the beginning of that sector to subdirectories. Yet later in the sector, where there's not even anything that matters, is where the defect is. Yet the drive won't give you the sector because it says, no, I can't read it perfectly.

Well, it turns out there's not actually any data there, in this case. But being unable to read that sector prevents you from following those two subdirectories, which could be huge, could contain all of your database data or all of anything. And so this is one of the keys to the miracles SpinRite seems to perform is this "give me the best you can" turns out to be incredibly valuable.

And so anyway, back then, and all of this technology I've managed to carry forward and keep alive over the decades, back then I built the most robust data recovery I knew how to build. I mean, just it does incredible tricks to get the data back, which is why it so often succeeds. And when all of that was done, when I had the data, I would then fix the low-level format and then go to the next sector. And as a consequence, SpinRite sped people's drives up.

Well, the other thing it did was it recalibrated them. It was a version of that lowering the drive a quarter inch each time and doing it again. And that was where the preventive maintenance part came in because, as your tracks are migrating over time, this migration caused by mechanical wear and tear, just the mechanical wear, and the fact that you're heating it up and cooling it down in this daily cycle, the tracks drift a little bit over time. If you run SpinRite before you can no longer read them, it rewrites them in - it reads them even though they're a little bit off-track. But when it rewrites them, it puts the directly under where the head is now to essentially realign the data. And so it might have a hard time recovering it. But once it rewrites it, now it can read it.

And the same is now true with error correction. Notice that these defects, if the tracks are migrating, the defects in the surface are not. They're staying still. But what that means is they're effectively moving relative to the tracks. So you might have had a defect that was in between tracks and not causing a problem, which migrates into a track because the track has migrated into it, essentially. And similarly, other defects might leave. So there's a lot more going on, for good and bad, inside these drives. A lot of this turns out to still be useful today. Interleave, that disappeared years ago. And I think it was with SpinRite 5 that I actually took that code out. It just wasn't useful any longer because all computers and drives got SMART and were able to handle reading every sector of a track in a single revolution. They were all 1:1 interleave. So that went away.

But all of that legacy excruciating reformatting data recovery and rewriting technology,

that has lived on and has ended up being SpinRite's legacy. It will almost always succeed in getting your data back. And so that's what SpinRite does and sort of why it's been doing it for so many years now.

Leo: Very awesome. You have a whole page on the website, don't you, that explains how SpinRite works. I mean, this isn't more detailed than that.

Steve: Yeah.

Leo: I don't think we'd ever really heard about the tilting hard drive exploit and the like.

Steve: Always like to bring some new details in.

Leo: All right. Let's talk about big - okay. So these are not, this is not my Linksys in the closet.

Steve: No. Although the Linksys in the closet is doing, in its own small way, very much what the so-called "big iron" routers are doing out on the Internet. So let's step back a bit and sort of - and refresh our foundation about how the 'Net works. Listeners who have not been listening for years will not have heard this before, so I beg the indulgence of those who have listened to every podcast for the last nine years, as we started the Year 10. I remember when I first sort of heard the rumors of an Internet, or the notion, like sort of science fiction at that point, of a global network. At the time we had modems, and you dialed your phone to connect to The Source or CompuServe or something, and...

Leo: [Simulating modem]

Steve: Yeah.

Leo: That was point-to-point, one-to-one. You didn't need a router.

Steve: Well, exactly. And I remember thinking, how can all the computers be connected together? I mean, how can that, I mean, because back then the notion was you would need wires to go between them all. And what we now know, what we now sort of take for granted is the incredible inspiration of the original designers, was this notion of what's called "packet switching," where a given computer would not need wires to every other computer, which was clearly impossible because then you would need "[n x (n-1)]/2" number of wires, and that's just not going to happen when "n" is large. Instead, you were essentially, you were able to reuse one pair of wires because the data that was going over those wires had addresses. Rather than the wire itself representing who you were connected to, the wire represented being connected to the world.

And then the brilliance was that the stream of data that used to just be a continuous connection between two endpoints, the stream was broken up into packets carrying the

addresses of the source and the destination. Essentially the addresses of those two endpoints were - the endpoints were not physical at each end of wires. They were logical as addresses in the packet. So it was an abstraction, a brilliant abstraction that made this work.

So then you have individual computers that all have their own wires connected, one way or another, to this thing called a router. And then the routers were connected to each other, and the routers knew who was connected where. And that's the key. So there's several levels of brilliance here. There's the notion of the abstraction of a connection is no longer the actual electricity. The electricity is now the data flow and the connection is a logical connection rather than a physical connection. Rather than an electrical connection, it's a logical connection represented by addresses carried by packets of data.

But to make this work, then, to bridge the logical to physical, something needs to know where an address is. We have the notion of addresses. IP addresses we're all very glib about, IPv4 giving us 32 bits of addresses. But we still need to know where because the job then, when one user puts one of these packets of data on their wire and sends it to their router, the one they're connected to, the router is like spokes hooked up to a bunch of other routers. And those routers have spokes hooked up to a bunch of other routers, ad infinitum.

But somewhere, some number of hops, as it's called, a hop is each of these links is a hop, some number of hops away is the destination. And when the packet of data arrives at the router, the router looks at the address, the destination, there's a source address and a destination IP, just using the destination IP. It sort of needs to look around at the links it has to other routers and decide which one to send the packet out that is sort of taking it in the direction of its destination. And if that happens enough times, finally it gets to the router that the guy that you're sending it to, the other end is connected to, and then it goes to them.

So routing tables is the way this is done. Now, the originators of this system, again with shocking foresight, I mean, for this thing to have survived basically unchanged, we've tweaked it here and there, but it's pretty much the way it's been. It's just incredible that it's gone from a research academic platform to this, I mean, to an amazing global thing, still basically the same. That's just, I'm just in awe of that. I'm sure that when they did 32-bit addresses, when they said how much addressing space should we need, they were laughing at 32 bits. Oh, we're never, you know, come on, are you sure you want to waste all those bytes on address space? Because we're never going to use up all of that space. Probably port numbers were the same way. Oh, don't you think - how many ports do we really need? And of course famously we have 64K ports because we have two bytes. And we have 4.3 billion IP addresses because we have four bytes, 32 bits.

So again, they designed for the future, even though they didn't know what the future was going to be, which, I mean, that's the mark of an engineer. They made some simple divisions of this space. Again, in understanding the challenge of routing, and this is why this is important, they said, let's define different kind of lumps. We'll have Class A networks that will be 16 million IPs. And, now, they were laughing, again. That meant essentially - 16 million is 24 bits. So what that would mean is, if you think about 24 as three bytes, three times eight, that would mean that the first byte would be the identity of the whole Class A network, that is, all networks beginning with one, and then anything in the other three bytes would be the 1 Class A network. And then if the first byte was a two, that would be the 2 Class A network.

And historically there have been, probably still are, I think HP is like 14. They still, because HP was in there very early, they're the 14 network. They've got all of the IPs

beginning with 14. Nobody else has an IP beginning with 14, and HP has all of them beginning with 14, a Class A network.

But since you only have a byte, that means there are some limits, too, because zero is reserved and 255 is reserved, sort of at the endpoints. So 253 out of the two - or 254 out of the 256 possible. And there are some ones, like 127 is famously also reserved. And 10, for example, the 10 network we all know, that whole Class A block is a private network. In fact, my IPs are all 10-dot here because it was just easy when I set things up. So what that means is that there is no public IP beginning with 10. That's all, that whole block has been reserved for use inside private networks. It's a so-called "nonroutable Class A network."

But the point is there are only - there are very few of those, only so many combinations of that first byte. So, again, these guys said, okay, well, we'll have some Class A networks. But we also want some Class B networks, that is, not everybody needs 16 million IPs. A lot of people who are smaller could get by with 64K IPs, that is, two bytes' worth of IPs rather than three bytes. So that's a Class B network where you have the first two bytes is the network number and then the second two bytes, the lower two bytes, the right-hand two bytes is which IP within that network.

So now, rather than having, like, everything in 14 network is HP, now you have things like the 24 Class A, the first byte is 24, and then the second byte is which one of the 24 subnetworks you're talking about as a Class B. And so you have many of the first bytes and then all of the values in the second byte creating Class B networks. And then they went one step further, and they said, and for really small networks, let's divide the B, the Class B networks into Class C networks, where a single Class B could be 255 Class C's, meaning that the first three bytes are the network number, and then the last byte, you know, one to 254, is the machine within that network. So a Class C network can have 254 different machines in it. So they set these scales.

Well, now, here's what's brilliant about that from a routing standpoint. The routers on the Internet, out on the 'Net, what is necessary for them is to know, to have a destination for every single IP address. That is, if any packet with any of those potentially 4.3 billion IPs arrives at a router, the router has to know where to send it. It's got to have a destination network. And if HP had its Class A network, all beginning with 14, then because of this hierarchy, because of the concept of networks containing networks, any packet that arrived at a router where the first byte was 14, the router didn't even have to look at the other bytes. It said, ah, 14 is HP. Send it to HP. And back then HP was much smaller than it is today, located somewhere on the peninsula in Northern California, and the bytes all went in that direction.

So my point is that this notion of networks and machines within a network, that dramatically simplified the job of the routing tables. The routing tables - imagine if there wasn't that kind of aggregation. Every router would have, I mean, if it was just completely randomly assigned, every router would have to have 4.3 billion entries. Well, it couldn't, but it would have to because every single IP would then be looked up to find out where it should go. And so that gives you a sense for the power of this notion of a network, the network bytes at the high end of the addressing space and then the specific machines at the low end. Because of this division, routing tables could be vastly smaller and much simpler.

Now, what began to happen as the Internet became popular is it became the case that companies, or the organizations, ARIN and the various Internet registries, started realizing that they needed to more closely match the IP allocations given to companies to the company size. That is, here was the problem. If you were bigger than a Class C

network of 254 machines, well, then, they'd have to give you a Class B. But that was 64,000 machines. And so the problem, there was no other easy granularity. So it would be possible to give someone two Class C's. But what they ended up with was something called CIDR, C-I-D-R, which is classless interdomain routing.

Essentially, what happened in, I think it was in '97, in the mid-'90s, was this A, B, and C distinction was broken, was deliberately dissolved so that instead of there being only byte-size boundaries, the routing architecture was enhanced so that the boundary between what is a network number and the machine number could then fall anywhere within the 32-bit space, meaning that this was classless. There were no more A, B, and C. It was, well, they're limited to powers of two so that you would go from, for example, 256 in what used to be a Class C, you could slide the boundary one bit over to the left and give 512, or one bit again and get 1024, or one bit again and so forth, to double the size of your network, rather than going to 256 times the size of the network.

Well, as all of this happened, routing tables grew. The architecture of routing pretty much held together. That is, an ISP would have a block of IPs. Then it would have lots of customers within its address space using those IPs. And it would have entries in every router out on the Internet for its network so that every router on the Internet, when an IP came in that matched the network number of this large ISP, that router would know which interface, which connection to send that packet to. And so that packet would bounce from router to router until it got to that ISP's network.

Well, the allocation of IP addresses is done through something known as "Autonomous Systems," AS numbers, ASNs, Autonomous System Numbers. And every entity on the Internet that is given a block of IPs receives an Autonomous System Number. Those used to be 16 bits long, but that's another thing that we outgrew. So that for some number of years now they've been 32 bits long because we outgrew the total number of entities that wanted a block of IPs. I remember when I was first getting serious about networking, Mark Thompson of AnalogX fame suggested that I register for an AS, that I get an Autonomous System designation.

And what that would have meant was I would have received a block of 256 IPs from ARIN. And they would be Gibson Research's IPs. And what's significant is that they would never change. That is, I could carry those with me wherever I went. It never seemed like that was something that I needed. I never pursued that. But as a consequence, I don't own my IPs. When I was with Verio, I had a block of IPs with Verio. I was briefly with XO. I changed IPs when I was within XO's umbrella, essentially borrowing from their IP space. And now I'm with Level 3 with a chunk of IPs that Level 3 has. So I've always been using the IPs of the ISP I'm in.

But many larger organizations don't. They did what I could have done, and that is applied for IP space and got it. So they are an Autonomous System with an allocation of IPs which are portable, meaning that those never change. And as they move, those IPs go with them. So that that means is - and this is part of what has happened over time which has caused a fragmenting of this otherwise sort of perfect routing. One thing that's happened is that invention of CIDR, the Class InterDomain Routing, has allowed allocations in smaller pieces.

But as those allocations were given Autonomous System designations, and as those Autonomous Systems moved to other providers, suddenly, when a block that was under a given network's umbrella and thereby could be routed with a single routing table entry, when that block moves somewhere else, now it can no longer be routed within that umbrella routing entry that gets the packets into that datacenter. It needs its own entry in every router on the Internet saying, whoops, here's an exception to otherwise this big

clock of IPs? Here's an exception to that. And for this smaller block you've got to send it over there.

Now, the way the routing tables work - again, it's a brilliant concept - is they perform what's called a "Longest Prefix Match." The prefix is those bits on the front from the left end which specify which network the packet goes to. And so the idea is, when you think about it, a short prefix isn't very specific. That is, there may be many networks that begin with 24, and then it's the next byte which specifies, like, what used to be a Class B network. It's more specific. And if you match all three bytes, the first three bytes, then you're really more specific. Now you're down to one of 256 machines in what used to be a Class C network. So the longer the match of the prefix, the more specific you get.

And so what the routers do is they look at an IP address of a packet coming in, and they, within their table, they find the longest routing table entry that matches, and that tells them where to send the packet. So what we want, in order to minimize the total number of entries in the global routing table, is what's called "aggregation." That is, we want a large ISP that has within its umbrella many smaller networks. We want that ISP, not to broadcast or, in routing terms, to advertise all of those networks because it's needless. If all of those little networks, subnetworks within a large ISP, are going to the same ISP, don't send or advertise them all separately on the Internet. Instead, just have one entry, essentially, that encompasses them all, so that the data can get to the ISP. And then the ISP can privately route them where it needs to.

So but the problem is, to the degree that that aggregation of smaller networks is not possible, we need individual entries in all the routers on the Internet, in so-called the "global routing table." And as I said, it's when autonomous systems that own blocks disaggregate from the original block provider, that it's then necessary to put an exception, essentially, in the table saying, whoops, sorry, here's a longer prefix match for this particular guy. Send it over here.

So over time, as we've talked about the whole IPv4 space depletion problem, over time the number of IPs, as we know, has grown and grown, IPv4 has grown and grown and grown and is now where - we're now at a problem where we're running out of IPs. This has created, not only IP growth, but in fact one of the contributors is that for a long time there were Class A-size networks unused. When we first talked about Hamachi, they were using the five-dot network because it was unroutable. It would have never been allocated. Nobody owned any five-dot IPs. And it was very clever for them to use that. But we then decided, whoops, we need to start using these IPs.

But the problem was, as we began using, for example, the five-dot network, we were chopping it up in small pieces. No longer were we going to give huge allocations to people just because no one was ever going to be able to use all these IPs. We knew that was no longer true. So right off the bat, over the last few years, all of the remaining unused space has generally been chopped up into smaller pieces than it was historically. So each of those - and those different pieces went to different people, different autonomous systems, so they all needed their own entries in the global routing tables.

So it's been a combination of the way the 'Net evolved, the tendency to allocate more smaller pieces as we began to run out of space, and then just the churn of motion of chunks of space disaggregating so that it couldn't be routed under a single umbrella and brought into a local domain where it could then be routed privately. But essentially, what used to be private routing had to be pushed out into the public because of disaggregation.

As a consequence of those things, we have been approaching a point where some routers

have a problem. There are a class of older Cisco brand routers. They are Catalyst routers. I had the model number, I thought, here, but I don't see it in my notes. Oh, yeah. They're the 7600 series routers, which are known as the Catalyst 6500s. They had a capacity of a million entries, yet the default is set to 512,000. So 512,000. In the notes here, I show today's global routing tables, how many entries there are. Now, not every global router has exactly the same number of entries because in some cases they've got a lot of networks which are local to them, so they don't need entries for their routers for their own networks. But other people need entries for those networks. So the actual number of entries varies from router to router. But they're right around 500,000. Generally about 500,000 is where we are.

Last Tuesday, something happened which suddenly pushed - and we've traditionally seen continuous, but rather slow growth. Last Tuesday an event occurred, and something pushed the global number of routing entries over the default setting for that particular older Cisco router, which could only handle 512,000. We're at 500 now, normally. And we're actually back to 500 now. It was just actually a mistake that was made by Verizon's Autonomous Systems, their ASNs 701 and 705. They aggregate a large chunk of routes. They have 72.69 and then all of the networks underneath. So that's what used to be a Class B size, 72.69.

What happened was they, by mistake, they pushed their subroutes out onto the public Internet. They disaggregated. What they would normally be routing privately, they pushed them out publicly, 170 additional routes appeared, and that caused the number of routing table entries to go above the number that that particular Cisco router could handle, and all, just in spotty fashion, all over the Internet those particular Cisco routers stopped being able to route. There's something called a TCAM, which is a - is it TCAM? Yeah, TCAM. It's a very fast associative memory, a content-addressable associative memory technology which allows these routers to essentially do incredibly fast, like line-speed lookup in order to process these tables. And it is settable, but the default is 512. With this mistake that Verizon made for a few hours last week, the entries went to 515. And when that happened, when those tables overflowed, the routers fell back to software lookup, which is vastly slower than hardware-based, virtually instantaneous lookup. And that's essentially what happened.

Leo: And Verizon immediately blamed Netflix for the whole thing. No, I'm just kidding.

Steve: [Laughing] Yeah. So Verizon realized they'd made a mistake. It wasn't their intention. It was just a plumbing screw-up where they published all these routes publicly that they intended to keep private. And that ballooned the routing tables. The routers sent all of these new routes out. Many of them, unfortunately, couldn't handle it. And so it created spotty outages, weird sort of effect throughout the Internet, where some people couldn't get to places, and some people could. And then as soon as those routes were removed, the network sort of healed itself and went back to being okay.

Now, Cisco anticipated this a few months ago. In May they sent out a notice to everyone who's got these routers they know of, saying, hey, you know, the global routing tables are approaching the default limit for these routers. What's interesting is, IPv6 entries take two slots, whereas IPv4 takes one slot. So right now the routers have a capacity of a million entries. About half of them, 512,000, are allocated. There's like a divider which divides that total space into IPv4 and IPv6. So 512 entries of IPv4. And because IPv6 takes twice as many, takes two slots, essentially, for its larger 128-bit address, those routers can handle 256,000 IPv6 entries. Right now the IPv6 table is only about 10,000

entries long.

So hopefully the people who have these routers last week realized their routing tables overflowed. All they have to do is just change the configuration, maybe to 768, for example, since IPv6 is still - the global IPv6 table is still so small that that threshold can be moved from the half point to the three quarters point. That'll give plenty of breathing room for IPv4 growth. And really, IPv4 can't grow much more. We're hitting the ceiling there, although it could continue to disaggregate, and so there could still be, even when we stop allocating new space, as chunks of IPv4 break apart and need their own table entries, they'll consume additional table space in the global routing table.

But it's looking like we've got plenty of time for IPv6. And hopefully we'll have less of a problem with IPv6 since its network, since it's got massively large network space, there shouldn't be nearly the same kind of disaggregation problem nearly as quickly for IPv6. So that's what's going on with the outage that we had last week, and some really cool information, I think, about the global Internet routing and how it works.

Leo: I'm just glad to know we can blame Verizon for the whole thing.

Steve: Ah.

Leo: Makes me happy.

Steve: There's a silver lining, Leo.

Leo: The only way it would be better, if I could blame Comcast. Wow. No, of course they didn't do it on purpose. It's a simple error, and it's really - it was going to happen sometime.

Steve: Yes, it was. As we've been creeping near this, I mean, actually, this is better that it happened this way, that it was like a little balloon that was a mistake, that was then able to be fixed. Because if this was the actual organic growth that truly hit 512,000 for the first time, there would have been no fix in an hour.

Leo: That's a good point, yeah. You need all those addresses, yeah.

Steve: But this was like, this is a perfect little wakeup call for those people who've realized, whoops, we need to take that note we got from Cisco in May a little more seriously.

Leo: Yeah. Is it a simple upgrade?

Steve: Yeah. It's just a matter of changing a number in a config file.

Leo: It's a firmware. Oh, it's not even that.

Steve: No, it's not even that. It's just...

Leo: Config file.

Steve: It's just the default is 512,000.

Leo: So there's enough RAM. It's not that the hardware can't handle it. It's just a config...

Steve: Correct. Correct. You simply change - actually, if you don't have an entry, it divides your memory in half, IPv4 and IPv6. And so all you have to do is add an entry to say I want it not half and half, but three quarters, one quarter. And then you're good, you're fine. Really, really interesting.

Leo: Yeah, fascinating. Fascinating stuff. Steve, as always, you're great. We have a visitor from Finland. He's a security guy for Microsoft Nokia. He does PKI implementations.

Steve: Cool.

Leo: Juha is in here. Or Juha, not Juha. Juha means something completely different.

Steve: Maybe that didn't translate, Leo.

Leo: Yeah, I think it's Finnish, you just wouldn't understand. Anyway, he's a big fan, enjoys the show. And he's...

Steve: Oh, great.

Leo: He's nodding along as you're talking, so I know we're all right.

Steve: Cool.

Leo: Thank you, Steve. You can go to GRC.com, that's where Steve hangs his hat for SpinRite, now that we know how it works, the world's best hard drive maintenance, and you can do a little recovery with it, too, utility.

Steve: Absolutely.

Leo: You also will find lots of free stuff. His work on SQRL is ongoing there. And next week, if god of the Internet permit, we will have a Q&A.

Steve: If there's no meltdown.

Leo: Which is getting increasingly more difficult to assume. So go and post your questions there about anything you hear, including the stuff we talked about today. That's GRC.com/feedback. Don't email him. You've got to use that form. You can also find 16Kb versions of this show for the bandwidth-impaired, beautifully written transcriptions by Elaine Farris, and all sorts of other stuff. It's GRC.com.

Steve: And the show notes this week with a bit of a freaky picture of Leo and Steve.

Leo: Yeah, get the show notes. And Tamahome has tweeted it. I retweeted it. So it's on the Twitter, as well.

Steve: Great.

Leo: On the Twitter you'll find Steve at @SGgrc.

Steve: Thanks, my friend.

Leo: And that's all there is to say. Thank you, Steve.

Steve: Talk to you next week.

Leo: See you next week on Security Now!.

Steve: Bye.