How the Internet Works, Part 1

Description: This week, after catching up with our usual grab bag of Internet-related security and privacy news, including another Microsoft Patch Tuesday, Steve and Leo plow into the first of a series of forthcoming episodes, which will be spread out over time, describing the detailed technical operation of the ever-more-ubiquitous global Internet.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-309.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-309-lq.mp3

Leo Laporte: This is Security Now! with Steve Gibson, Episode 309, recorded July 13, 2011: How the Internet Works.

It's time for Security Now!, the show that brings you the latest security news, with this guy right here, Steve Gibson, of the Gibson Research Corporation, GRC.com. Steve's a great man, a guy who's been doing this show now for six years. And of all things, after six years, we're finally going to find out the one thing we probably should have started with: how the heck the Internet works.

Steve Gibson: Well, I mentioned some time ago that we were going to do this series. And I know it's months ago because, as you know, Leo, as our listeners know, we've had a very busy last few months with the addition of the Attacks & Breaches category in the top of the show, and just never, I mean, like, literally never a moment to breathe. And so people have, through Twitter and through the mailbag, GRC.com/feedback, have been - I see a constant question: When are we going to do the Internet, how the Internet works series? So we're not going to suspend, we can't suspend everything else to do this. But I wanted to start today. Today just sort of seemed like the right day to start the series which will very carefully and methodically and in sufficient detail that I think it will be very interesting to people, we're going to really understand all of the fundamental concepts and technologies that is the Internet, that is the way it works.

And today, Part 1 of the series, in sort of setting this up for myself and thinking about it, I realized there were three things, just three, three things that back then, at the time, had never been done before, completely unproven. They were new concepts. And there's just three things that are really responsible for everything else that followed, three things that the original designers of this thing that outgrew its roots to become a global network, they got these three things right. And so we're going to talk about the three key concepts that underpin the way the Internet works as our topic for today.

Leo: Wow, three things, that's all it took.

Steve: Yup.

Leo: I can't wait to find out what they are.

Steve: Three things done right.

Leo: All right. If you're playing the home version of Security Now!, try to think of what those are, and we will answer that question in just a bit. We also have, of course, security news. There's a big update a couple of days ago.

Steve: Yup, and a scary revelation from Microsoft about something that had been wrong since Vista SP1...

Leo: Oh, that's nice.

Steve: ...that no one found out about until they fixed it.

Leo: Isn't that special. So Patch Tuesday was Tuesday; right, Steve?

Steve: Yes, it was indeed. Now, there were 22 different security flaws of various ilk fixed. 21 were nothing, were not a real big deal. They spread across various products, pretty much all of the things. And they were the so-called "privilege elevation" problems which are a concern because we do rely on administrative rights being godlike within our machine, and regular user rights being restricted. So it is the case that that barrier is good to have, and we don't want it to be penetrated. So there were 21 different ways of essentially getting around that kind of barrier.

But the 22nd problem was interesting. Microsoft revealed at the time of this patch being available that there had been, since SP1 of Vista, which is when they introduced an upgraded Bluetooth stack, as it's called, a protocol stack for Bluetooth - and we'll actually be talking about protocol stacks relative to the Internet by the end of this episode. There was - and this is like the worst of all possible nightmares. We're assuming that no one knew about it. But this was a buffer overrun-y sort of problem in a wireless protocol. So quoting from Microsoft, they said:

"A remote code execution vulnerability exists" - now, remote code execution vulnerability exists - "in the Windows Bluetooth 2.1 stack due to the way an object in memory is accessed when it has not been correctly initialized or has been deleted. An attacker could exploit the vulnerability by constructing a series of specially crafted Bluetooth packets and sending them to the target machine. An attacker could then install programs; view, change, or delete data; or create new accounts with full user rights and take complete control of a system without any user notification at any time."

So, and then in the FAQ and the Q&A, people were saying, well, if I don't have Bluetooth installed, then I'm safe; right? And it's like, well, no, because Vista and Win7 will instantiate the Bluetooth device if you temporarily give it, like, a removable Bluetooth over USB. So somebody could come along and stick a little Bluetooth dongle into a machine that has never seen Bluetooth, and Vista and 7 will happily bring the driver online. That's where the problem is. And then an attacker could get into your system. And of course...

Leo: So they don't really need access to your system. They just have to be within Bluetooth range of your system? Is that the case?

Steve: And they don't ever have to be associated. We've talked about how Bluetooth protocol works, where there's a whole pairing association which is the one moment of vulnerability with Bluetooth before a common secret key is negotiated between endpoints, which are then memorized by each endpoint so you never have to go through that again when you're reconnecting to a device that you have once paired with. This doesn't require pairing, doesn't require a connection.

Basically it's a breach in the lower level infrastructure of the Bluetooth driver. I mean, it's really what you don't want. And so we're, like, hoping we don't have these in the network, the regular wired network stack that we're all using and depending upon pretty much 24/7. This is that same thing in Bluetooth. And many machines just run with Bluetooth on. And in fact there's some feature that Microsoft's added, again to make things easier, where if you're in, like, network discovery mode, it will turn on your Bluetooth radio without your asking it to, just in case maybe that's what you're trying to discover.

So anyway, the good news is this has been fixed now. It's an important update, more so I think than most things, because now the bad guys know about this, and we know that they're on the hunt for it. We see this all the time, is there's now a window between disclosure and the time when people are going to get patched. And we know from history that lots of machines are very late in getting themselves patched. So one could imagine that the hackers are all focused on this problem now. So I'm sure our listeners will do what they can to get themselves up to speed with this.

And Microsoft does offer in their Knowledge Base article a command which I didn't bother to write down and save because it's too long. But there is a way, if you, for example, have Bluetooth hardware in your system, but you absolutely never ever want to use Bluetooth, and you're sure, and this is reversible anyway, there's a command you can give that essentially permanently shuts down Bluetooth access. It's a net shell command with a bunch of parameters, where you disable...

Leo: Probably deletes the driver; right?

Steve: Yeah, it disables the driver so it cannot be started by Windows. And if you really never use Bluetooth, that would be a great thing to do. You can get yourself a little caught up, though, or caught out. In the old days I used to always disable the DHCP service in my Windows systems because I don't use DHCP within my own network. I've got all my IPs assigned statically. And then I'd take my laptop or a machine somewhere else, and it would refuse to get on the Internet. It's like, what's wrong with this? And then I finally would remember, oh, wait a minute, I disabled the DHCP service because

I'm wanting to run no services that I don't need.

And so you do want to remember, if you disable Bluetooth in this fashion, that you did so, and remember to reenable it if you should ever need it in the future. But really, turning the radio off at least is a really good thing. And if it's a laptop, it saves battery power. It's not using much, which is the whole point of Bluetooth, but it does save some. And it didn't affect anybody on XP [pointedly clearing throat].

Leo: So this is something that they kind of added.

Steve: Yes, in SP1 of Vista.

Leo: A feature of SP1.

Steve: Hey, it's a nice feature. Vista, from SP1 on, the Bluetooth stack was at v2.1.

Leo: That's important because it underscores the fact that you can add bugs. Just because an operating system's been updated doesn't mean it's bug free. You can actually add bugs, add exploits, add holes.

Steve: Oh, Leo, how many times do I pound our listeners over the head with I'm still on XP. I never was subjected to this. Nor were any of my laptops, because I've stayed there. Which, by the way, has exactly 999 days remaining before Microsoft...

Leo: Yes, the countdown began yesterday.

Steve: 999.

Leo: I think you should put that on a website somewhere. That's about three years, folks. We're all right.

Steve: Yeah, we're okay. There was a minor fix to Firefox v5. It's funny, I've seen a lot of grumbling about Firefox. They've already got 6 in beta, Leo. And it's like, wait a minute. We just had 4. And I worry...

Leo: And 5.

Steve: I worry that they're just trying to play the version number game because...

Leo: Well, I think what they did, and this is what Chrome has done, is that they kind of - and they've even said this explicitly. Version numbers are no longer as important

as they were. They don't reflect this - in the past, you and I, we've been around long enough, we remember that one whole version update is considered a major update. And then the point updates are considered minor updates.

Steve: And also sometimes the major ones are not free, but the point ones are free. So there's often that kind of differentiation.

Leo: Yes. But Mozilla has said explicitly, we're not going to do that anymore. And I think it's Chrome that really drove that because Chrome is pushing - they're up to 11, and they're just pushing versions all the time.

Steve: In fact, Chrome's version numbers look a little bit like the U.S. debt clock.

Leo: They're spinning.

Steve: So we had a v5 update for the - only affects the Mac because there were two problems that were found on the Mac. There was a Firefox v5.000000 problem, where it itself would crash, which they have fixed. And then they also found a conflict with Apple's own Java for Mac, which could prevent the Java plug-in from loading. So both of those are fixed. So only Mac people, if you're on v5, then you may want to fix that. Or I imagine Firefox is probably bugging you to fix that, which is...

Leo: Funny, because I just went from 4 to 5. All right. Okay.

Steve: So in security news there's a new problem on the horizon. Kaspersky discovered a new trick being played by rootkit malware. They've got one that they found called 'Cidox." And we're all familiar, because we've been talking about it now for several years, with so-called MBR patching, the Master Boot Record. The master boot record is the first sector of the hard drive, which contains a partition table. And as we have said before, the partition table is a relatively small little bit at the end of the 512-byte master boot record sector. All of the other space is actually executable code. And it's that code which reads an entry from the partition table to determine which of the four partitions, typically, is marked as bootable. And then that code reads the values from that table to determine where the OS lives on the balance of that hard drive.

So this is executable code. The master boot record is a little program that runs which itself reads the partition table. So that opening means that you can use that trick for good or for evil. TrueCrypt uses it, arguably, for good. That's the way TrueCrypt is able to give us our whole drive encrypted, is that they modify the master boot record to jump to some space in the balance of that first track because, by convention, the first partition of the drive always starts at the end, well, it always started on an even track boundary. Since we've used, I was going to say "cylinder," but that's not true. The end of a partition is on a cylinder boundary. The beginning is at least on a track boundary. So since we've already taken up one sector of the first track, we can't have a partition on the balance of that. We start the partition at the beginning of the second track, leaving all of the track free.

So TrueCrypt uses that to install their own little "prompt the user for the password" and "decrypt the bootable partition." And it also hooks Interrupt 13, which the operating system uses in the BIOS in order to get itself booted just enough to get the TrueCrypt Windows driver, or Mac or whatever driver, running. And then it gets you going. That's the happy way of using a modification of the master boot record. The bad way is what rootkits use now, increasingly, which is they'll install themselves there in order to get a foothold into the operating system and inject themselves prior to it starting up.

Well, it's become so popular that more and more people are checking for it now. It's more difficult for malware to install itself in that first track of the hard drive by modifying the master boot record. So they've come up with a new means of gaining a preboot foothold. And this Cidox malware that Kaspersky found does that. It takes advantage of the fact that there is a period of time after the master boot record has done its work and loaded the first file of the operating system. That first file then turns around and begins to pull other pieces together. But the OS is not booted yet. There's a lot of work that has to be done before you actually boot the operating system.

This Cidox malware, a new form of rootkit which unfortunately is now going to naturally become prevalent, it recognizes that there's this, like, more opportunity to get up to mischief than just modifying the master boot record. It places a chunk of its own code in unoccupied space in the NTFS partition. And it's only able to play this game on NTFS-formatted partitions. That is, it won't work if your OS was FAT32 or FAT16 formatted. But no one's is these days. It's going to be NTFS. So it puts its code in unoccupied space.

Then it overlays part of what Microsoft calls the IPL, the Initial Program Loader. And that's the code which parses the Master File Table, which is one of the management tables on an NTFS partition. The NTFS IPL parses that master file table in order to begin to understand the structure of the disk in order to sort of create a preboot file system interpreter, which is then able to go find the other pieces of the operating system that it needs to load. By subverting that phase, it's after the master boot record, but before Windows has loaded. So this code takes the chunk of the IPL that it overwrote, encrypts it, and adds it to the end of itself. So it's able to get control at that point and load itself in place of one or two of the normal Windows drivers and then continue to do the job that the initial program loader would have done in order to bring Windows up.

So essentially it's subverting a later stage boot portion of the Windows process, which nothing is currently looking for. So we've got a new means of subverting the system. And when you think about it, there is. There's, like, we're sitting around waiting for Windows to boot. There's a large window of opportunity. I think, now that this has been done, we're going to be talking about these kinds of things. And we're going to have to have some sort of new form of protection because just preventing rights to the first track, that's the other thing that's happened is we know that there are BIOSes, for example, which make the master boot record read-only. They protect that first sector because it's vulnerable. This is able to install itself even on such systems.

So we're going to have to have a next, in this eternal cat-and-mouse game, a new way of protecting ourselves until the operating system is able then to protect itself because, during the time when the bits and pieces are coming off the disk, it's actually not alive yet. It's not conscious. It's just there, just being dead code. And bad stuff has a chance to get in then.

Leo: So the kernel protection that Windows 7 64-bit has, for instance, won't stop this.

Steve: Precisely. It doesn't exist yet. It hasn't...

Leo: It hasn't run yet. You need the code to be running.

Steve: Exactly, exactly.

Leo: All right, yeah, it makes sense.

Steve: Now, you may have run across this news story. I saw lots of interest in the media, and also just from our own listeners, about this question that's being raised about whether the court has the ability to compel someone to give up their passwords. And this turns out to be a constitutional issue. This is the Fifth Amendment which protects us from having to incriminate ourselves. A woman named Ramona Fricosu in Colorado is in the middle of being prosecuted by the Justice Department for her alleged role in a mortgage scam. Fricosu had an encrypted laptop which was seized from her bedroom during a search of her home. Federal prosecutors want her to decrypt it so that they can see its contents.

Her lawyers argue that compelling her to do so would be self-incrimination. It would be violating her Fifth Amendment protection against self-incrimination. The government says that its request is like asking for a literal key for a literal safe that may well contain incriminating documents. And prior law and Supreme Court rulings have said that you can compel a person to give a physical key to a physical safe. And apparently the question turns on whether the key is in your mind, whether or not you're able to compel it.

Leo: That's interesting. That's really interesting, yeah.

Steve: Isn't that interesting? Yeah. There was, on NPR, Declan McCullagh, who is [CNET's] chief political correspondent, did some research. And he said: "If you have encrypted files, is that like a safe? You can be compelled, according to Supreme Court precedent, to turn over the key to a safe. That you can be compelled to do. But you cannot be compelled to turn over the passphrase, or, that is, the combination to the safe. So there is existing law that says you cannot be compelled to turn over the combination if it's in your mind, but you can be to turn over the key. Now, of course the EFF has weighed in. They've put together an amicus brief to the court with their position. And they have a blog posting that said that the EFF urges court to uphold privilege against self-incrimination: "Prosecutors Demand Laptop Password in Violation of Fifth Amendment." Their blog post says:

"The Electronic Frontier Foundation (EFF) urged a federal court in Colorado today to block the government's attempt to force a woman to enter a password into an encrypted laptop, arguing in an amicus brief that it would violate her Fifth Amendment privilege against self-incrimination.

"A defendant in this case, Ramona Fricosu, is accused of fraudulent real estate transactions. During the investigation, the government seized an encrypted laptop from the home she shares with her family, and then asked the court to compel Fricosu to type the password into the computer or turn over a decrypted version of her data. But EFF

told the court today that the demand is contrary to the Constitution, forcing Fricosu to become a witness against herself."

And then they said, quote, from their amicus brief: "Decrypting the data on the laptop can be, in and of itself, a testimonial act - revealing control over a computer and the files on it. Ordering the defendant to enter an encryption password puts her in the situation the Fifth Amendment was designed to prevent: having to choose between incriminating herself, lying under oath, or risking contempt of court," all which the Fifth Amendment protects us from.

"The government has offered Fricosu some limited immunity in the case, but has not given adequate guarantees that it won't use the information on the computer against her. 'Our computers now hold years of email with family and friends, Internet browsing histories, financial and medical information, and the ability to access our online services like Facebook. People are right to use passwords and encryption to safeguard this data, and they deserve the law's full protection against the use of it against them,' said EFF Staff Attorney [Hanni Fakhoury]." And they said, "'This could be a very important case in applying Americans' Fifth Amendment rights in the digital age.'"

Leo: Wow. This is why law is such a fascinating pursuit, because both sides have - there's merit to both ways of interpreting it.

Steve: And I love the language: The act of divulging the password is a test, in itself is a testimonial act. And so they clearly wrote this to strengthen their position that you cannot compel testimony. And if revealing the password is a testimonial act, then you cannot [compel] the password to be revealed, which of course we've all talked ourselves blue in the face about encryption and passwords and all that.

Leo: I thought they could ask for your password.

Steve: Yeah, interesting.

Leo: It's good to know that they, I mean, it doesn't mean they can't ask. But you have the right to say no. They may well ask. In fact, I know they will ask.

Steve: And you can say no without being held in contempt and put in jail for saying no. And that's the point.

Leo: Right. It's good for us all to remember. You don't have to reveal it.

Steve: I thought this was really interesting, too. A buddy of mine in the U.K., Paul Byford, who I know as Sparky in our newsgroups and has been a super great contributor for years, sent me a little blurb from the U.K., since that's where he is, about the News of the World phone hacking scandal. It's been getting a lot of attention here in the U.S. because the owner of the News of the World newspaper is of course very well known to us. He's the owner of the Fox Network.

And what Paul picked up on was the nature of the hacking, which is the reason I thought our listeners would find it interesting. All it was, I mean, this was no big, deep, dark, 007 counter-cyberterrorism-style stuff. Get a load of this, Leo. When people were having cellular phone service established, wireless phone service, they had the ability to get their voicemail, not only from that phone, which identifies itself to the voicemail service, but maybe if they wanted to pick up their voicemail, and they were away from their phone, they had the ability to get it from any other phone using a PIN. And the default PIN was either...

Leo: 1234?

Steve: 1234...

Leo: Or the last four digits of your social.

Steve: Actually, or 0000.

Leo: In the U.S., just in case you want to hack some phones, it's very often the last four digits of the phone number itself. That makes it nice and easy to remember.

Steve: Well, of course you would have to know the Social Security number of the person you're trying to...

Leo: No, not SSN, the phone number.

Steve: Oh, the phone number. Oh, goodness. So this has been changed. But for many years the de facto other phone access PIN was either 0000 or 1234. So all these reporters who worked for the News of the World were doing is, if they were able to get the phone number of the celebrity or state official in question, they would simply hope that that person had not changed their PIN. And oftentimes they were never changed. In fact, that feature was never even used because they were always just using their cell phone to get their voicemail from it, and it identified itself to the carrier.

So this is another classic security glitch where I would say this is one of those "disable anything you're not using." And so if you're not using non-cellular phone access to your voicemail, if in any way possible, disable it. Otherwise, change the PIN and forget it, forget what the PIN was. And then it's effectively disabled. Although there's only, what, 10,000 combinations. So if these guys, these reporters were really determined, and as far as I know there's no password lockout on PINs for voicemail access, you could just sit there and try them all.

Leo: It's really an ethical violation more than anything else. They apparently were also using pretexting and other even more intrusive social engineering techniques. So this is pretty nasty.

Steve: It was bad. So, Sony, once again in the news for how to completely misunderstand the whole intention of a CAPTCHA. We've talked about a CAPTCHA, how the idea is it's attempting to determine whether you're human. And so unfortunately the technology has escalated to the point where it is sometimes difficult to convince the website that you are human. You are, I am, our listeners are, Leo. But there are CAPTCHAs where I just scratch my head, and I think, okay. What the hell is that?

Leo: You almost feel like, gee, I wish I could have a computer help me solve this one.

Steve: I guess I'm not human.

Leo: Yeah.

Steve: So the problem is they've become very difficult. But the goal is that you don't want some automated machine which brings up the web page sort of in its own belly and then wants to post to forums. These are used for creating accounts, for posting to forums, for any time you want to prevent spam-ish sorts of behavior. You create a puzzle which hopefully only a human can solve.

Well, Sony didn't really understand that when they designed the CAPTCHA for one of their professional media websites. The CAPTCHA functions, in this Sony case, by providing the text which the user's supposed to enter in the page's HTML, which...

Leo: [Laughing]

Steve: Oh, I know.

Leo: Well, which means it's always the same; right?

Steve: Well, no, no, it changes.

Leo: So they have, like, a choice? A case statement?

Steve: Well, the point is the computer can read that.

Leo: Yeah. It is reading it.

Steve: Yeah, it loads the page and gets the secret.

Leo: That's called rendering a page.

Steve: Because then what Sony would do is, they used some JavaScript to algorithmically distort the visual presentation...

Leo: Oh, no.

Steve: ...of the text that was already in the HTML, but not visible to the human. So the human had to go through all the trouble of figuring out what this distorted text was, whereas the computer had the undistorted text that was just in the plaintext, just in the HTML of the page. And in fact the human could have done a view page, view source, to see it themselves and then typed that in. That would have been easier than having to figure out what this thing looked like. So, yeah, Sony, busted once again. Didn't quite have the right idea there.

Leo: Boy, it's almost like Sony, like, believed in hiring the worst possible security people. It's like it's an example of how to do it wrong.

Steve: Yeah, I just think they just did it ad hoc. They just, every instance, they came up with their own solution. And unfortunately none of them are good.

Leo: Amazing. It's like...

Steve: So I had a bunch of interesting feedback from Twitter in the last week. Nate Allen said, "Started reading 'Daemon' on your recommendation. Amazing book." Brandon Bodnar said, "Thanks, @SGgrc. I found my new favorite book on Audible. 'Daemon' by Daniel Suarez is great. Can't stop listening." And then he mentioned the domain "thedaemon.com." And then FuerstOpus, who's a guy in Oklahoma, said, "Re 'Daemon,' for years Gibson Research has been listed as a security link at thedaemon.com. You may have been a resource for the book"...

Leo: Oh, yay. Daniel Suarez is a fan.

Steve: ...he tweeted to me. So I thought that was an interesting kick. @mStyle, whose name is Robert Altman, said to me through Twitter @SGgrc, he said, "300 hours and still going. SpinRite at 58% complete, 7 not completely recovered sectors, 1 unrecoverable. SpinRite is amazing. Nothing else would touch this disk." So, nice to have a little feedback. It is making some progress. And I guess if he's willing to wait for it, it's better than not having anything else be able to recover it.

Troy P. Peterson tweeted that Chrome now blocks insecure scripts on HTTPS by default, and gave a link to a Google blog post. And that's interesting. I checked it out. What Chrome was doing, Chrome has essentially upped the ante on protecting users from themselves. We've all seen sites that give us what's often called a "mixed content message," where, for example, your HTTPS is in the URL, so your main page content was delivered over SSL securely. But then, as we know, a page almost always contains links to other assets, which your browser then, after parsing the page which came in over SSL, the browser then looks up all the other things - images, other bits and pieces, and maybe script, maybe JavaScript - which it then brings in.

So it used to be that until Google strengthened Chrome in this way, if any other fetches from the page were HTTP, that is, not over SSL, you'd get a little warning that said there's some things on this page that are not secure, do you want to proceed? Most users just click right through that, saying, yeah, well, I want the page, and right now it doesn't look so good because I'm not getting the rest of my stuff. Well, one of the things that could be delivered was the page's script, which as we know is very powerful. And if the page was delivered securely, we absolutely want the scripting content to be delivered securely. So Google Chrome, as of this most recent update, no longer even gives us the option of saying, go ahead, I want insecure scripting. If the page is delivered secure, Google will now block nonsecured scripts from being delivered, which, again, that's another nice step forward for Chrome.

Paul Morgenthall, who is on Twitter @fuzzymuzzle, said instead of a plus sign - he's referring to Gmail, adding names to Gmail that we've referred to recently. First it was suggested, then it was de-suggested because the plus sign confuses so many email clients. He says, "Instead of a plus sign, add dots to a Gmail address." So it turns out that works. And he gave a few examples. He says it's less descriptive, but always accepted. And what's interesting is that it's only embedded dots. So, for example, you could add dots anywhere within, like, between the first letter and the last letter of your Gmail address. And you can even multiple dots at a time, and they're just discarded. I mean, it doesn't create a unique account. And so that is a way to add accounts to Gmail and then to provide some discrimination that a user can generate ad hoc. And as we know, there's lots of ways you could salt dots in between the letters of your email address, depending upon how many letters you've got.

Leo: It's fundamentally different, though, because the dots are ignored.

Steve: Right.

Leo: Anything after a plus is ignored. So there's a difference there. And, yes, you have to use some sort of cryptic combination of dots to modify your email. With a plus you could actually use a descriptive word...

Steve: Which would have been very nice.

Leo: Yeah. And I'm surprised. I would think any modern email system would ignore whatever's after the plus. But I'm not - not email system because it's not the email system. It's always going to Gmail. It's the site that you're giving this email address to that doesn't like the plus.

Steve: Actually, it works several ways. There are actually, from all the feedback that I got, there appear to be email servers that to this...

Leo: Oh, that will reject the mail, I get it. They won't pass it on.

Steve: That to this day, yeah, they just won't forward it. They go, this is not - I mean,

and these are dumb. But unfortunately it's what we're dealing with today.

Leo: I've used plus for a long time, and I haven't run into anything. But in most...

Steve: Oh, that's interesting.

Leo: ...cases when I use it, it's on a site that I don't care for their email anyway. So, I mean, I may be missing something and not know it.

Steve: And finally, a very good security tip from Dale Perkel, who is DeepPurple77 on Twitter. He said, "A great tool for security people, Qualys BrowserCheck, shows which plug-ins have security issues." So this is, it's browsercheck.qualys.com. So browsercheck.qualys.com. And I put it into mine, and it wanted to install a plug-in. It's like, I don't think I'd want to do that. But it also allowed me to perform a quick scan on the spot, and it found three things, three Firefox add-ons that I had that were not - that it said were not secure. And not surprisingly, two of them were Adobe, and so there were newer versions of both of those. So it's like, oh, hey, this is good to know. So browsercheck.qualys.com.

Leo: I'm not familiar with that. I'll check it out.

Steve: I wouldn't be surprised, my friend, if it pulls up some red marks.

Leo: Uh-oh.

Steve: And I did want to share, oh, I wanted to share two things. First of all, a bunch of our listeners, I was really pleased and impressed, a bunch of our listeners volunteered to pay for that copy of SpinRite that Ryan McCain got from me for free after the horrible Katrina event that he and his family suffered. And, I mean, I don't need anyone to pay for it. I was more than happy to let him use a copy of SpinRite for free. But I just wanted to thank all of our listeners for their kind thoughts and sentiments. And it was quite an outpouring.

Leo: And I was not surprised to hear it.

Steve: Yeah. And Russ Palmeri said, "Dear Steve, please add my name to your long list of happy customers. My laptop, which I need tomorrow to teach my anatomy and physiology class, would not boot up properly. It took 20 minutes to get to my desktop, and everything I tried to do had a three- or four-minute delay. The computer was unusable. I called Dell, and the tech walked me through the Dell diagnostics. The hardware all passed, including the hard drive. I was unable to use System Restore. It crashed. The next step was to reinstall Windows and, of course, all my programs." And actually, Leo, that wouldn't have worked, either.

But, he said, "Fortunately, my data was backed up, mostly. Still, I had hours and hours

of drudgery before me. Before taking that drastic measure, I thought I might try the SpinRite utility. To be honest, I bought SpinRite as a way to thank you for your outstanding Security Now! podcast which has brought me so much enjoyment. I was afraid it might require more expertise than I had to run it. But the program is very well designed to guide the user through the necessary options.

"I booted up from the SpinRite CD and went to the recovery choice number two. Then I left it alone. I came back later and saw one of the sectors was bad. SpinRite had fixed it. SpinRite chewed on that for a bit and then was finished. I booted up without the CD. Windows ran chkdsk by itself, and up came my login immediately. My laptop is now working perfectly. When I took math in college I learned that some theorem proofs are so beautiful, they're called 'elegant.'" He said, "Well, SpinRite is an elegant utility. It sure saved me tonight. Thank you so much for Security Now!, and thanks for SpinRite. No one who owns a computer should be without it. Russ." So thank you very much, Russ.

Leo: Thank you, Russ.

Steve: And of course I agree.

Leo: What a surprise. All right, Steve. We're ready. How doth the Internet work?

Steve: Our listeners have been patiently waiting for 48 minutes, while we've caught them up on all of the news of the week, about what are the three key concepts that I would argue everything about the Internet is based on. These were new at the time. And really they were so radical that it was - it didn't necessarily surprise the engineers when these things worked. But, boy, it would surprise them today that they are still working.

And that, I think, if there was anything I would say which demonstrates the power of getting it right, it's the fact that the Internet, the fundamental core of the Internet is virtually unchanged since it was first conceived of and brought up to speed. Leo, we don't have any computer technology that is, I mean, maybe hard drives are still around, and they have matured dramatically, I mean, so that they're unrecognizable, essentially, like in terms of density and so forth. You could not use today a hard drive from that era because those were disk packs that looked like washing machines. I mean, so I wouldn't say that stayed the same. But these three key concepts are the reason the Internet is the Internet and has survived. Virtually, these are unchanged. One is packet routing. The second is best effort is good enough, which is a huge breakthrough concept.

Leo: Yeah, I love that.

Steve: And the notion of a protocol hierarchy, that is, a very careful and clear organization to the structure of the data that moves across the Internet. So we're going to look at each of those three in turn.

When the Internet was being created, when it was being designed, the way computers were interconnected was called either a "leased line" or a "switched circuit." But both of those worked the same way. A leased line was simply a connection which you leased typically from your long-distance carrier that I think even back then was AT&T or Bell. And, for example, the major government institutions and our military complex and

educational institutions, where they had, I mean, they were bringing big computers online, and they were wanting to interconnect them. So they used leased lines, point-to-point circuits. They were typically 64K baud was the operation speed of these things, so not super fast, and generally reliable. But the whole concept was a point-to-point circuit.

So as the number of computers increased and the computers wanted to interconnect to each other, it became increasingly expensive because these circuits were not inexpensive, and they were also cross-country in many cases - Stanford wanting to connect to MIT. And there was a Stanford-MIT connection that tied their machines together as we were sort of beginning to experiment with networking. And of course all of us old-timers remember AOL and CompuServe, The Source...

Leo: GEnie, The Source.

Steve: Exactly. And these were point-to-point connections. Somewhere there were massive banks of modem pools, and all of our machines had modems, and we used our phone line to dial into these services in a point-to-point circuit, to create a point-to-point connection from us to them. And that's the only thing that circuit was doing during this time. It was dedicated to us. And of course that caused problems when somebody else wanted to use the phone. Oh, I'm on the computer. So you had to wait. And you also had to suppress three-way calling because, if another call was coming in, it would briefly disconnect you and then drop your connection. We remember those days, as well.

So, I mean, so the system worked. But it was not efficient. All of that was called a "switched circuit approach." So it represented a fundamental rethinking to say, wait a minute, instead of creating discrete point-to-point connections between everything, we're going to break these connections up into packets of data. There was no notion of a packet. That didn't exist. You established a connection, and you put data - whatever you wanted to - in one end, came out the other. So there was no packetization. There was no discrete packaging. So that was a conceptual breakthrough.

And then what happened when you had packets was you thought in terms of sharing resources. And that's, when you think about it, that's really the big difference in the Internet is, you know, a cable modem is hooked to a cable which is a shared resource. Lots of people in our neighborhood have their cable modems hooked to the same, physically the same electrical connection, and we share that resource. All of the high-speed links we have now running around our cities and interconnecting our cities are shared resources. And so it's this sharing which you can only get if you drop this notion of a point-to-point switched circuit, and you somehow allow a mass of data to find its way among the shared linkages. That's this fundamental rethinking of the way data gets moved from one point to another.

So the routers which exist at various points throughout this internetworked architecture and the links between the routers are being shared by all of the data which moves across them. And they're sort of a - they're owned by different organizations which end up being compensated for providing the resource and the bandwidth links, so that the system is self-sustaining. But it's also - it sort of changes the dynamics because the second part of this huge revelation in the way we interconnect our computers is the concept that "best effort" is enough. And that was a huge one because, when you used to have this whole switched circuit, this whole point-to-point thinking was you needed the link to be up and the link to be reliable. And anything, as long as it was up, anything you put in one end came out the other.

But the designers of this packeted approach, instead of just having literally like a phone conversation which is continuous, they chopped up the data into packets and weren't even sure what route it was going to take. We'll be talking about routing extensively in a future installment of this "How the Internet Works."

But the idea was that instead of this continuous connection, you packetized, as I said, and then sort of launched the packet off and hoped for the best. The reason you have to hope for the best, the reason there isn't a guarantee - and this, again, it's like, if I can try to convey the incredible change of thinking that this took, is that you have a router that will be connected to a number of other routers, and those routers are connected to a number of other routers. Well, each link has a bandwidth which it's able to move data. And they don't all have to be the same. You could have some high-bandwidth links and low-bandwidth links. Especially in the beginning, it was a very heterogeneous, you know, all things were different, network that was built.

So packets, these individual packets of data are arriving across some links to routers that then inspect them - and we'll be looking in detail at how that happens and what it means - and then determining where they should be sent or forwarded to. But you have a fundamental problem because what if, for example, more packets are coming in than the router is able to send out? Because that could happen. You could have a burst of packets coming in over one or more high-speed links which are all saying, we want to go out over there, out of a low-speed link. But the router can't send them out at the same rate that it received them in.

So, I mean, and there were people arguing in the beginning, well, this won't work. Obviously this is crazy. And so the engineers thought for a while, and they said, well, we'll put some buffers in the routers, and packets can wait in an outgoing buffer until there's time. So that sort of - that solved part of the problem because, if packets came in in bursts over the high-speed link into a router, it would check each one and assign them to their outgoing interfaces and put them in a buffer to be sent. And so the interface hooked to the link that's going to some next router, it would have a buffer that could hold a certain amount of data. So that could solve burstiness problems, where high-speed bursts could come in and then queue up in this outgoing buffer on its way to the next link.

But what if you just kept getting a lot of more data in than you were able to ever squeeze? I mean, the buffer would have some finite capacity. So it was entirely possible - and again, this is the genius of the designers. They said, if the outgoing buffers overfill, if the buffer is full so it cannot accept a packet to be forwarded, we're just going to discard it. We're going to say, oh, well, too bad. So, and again, during the negotiation of this technology, the people said, well, wait, what do you mean? You're just going to throw that away? But we sent it. And the designers said, yes, we know. And...

Leo: You did your best.

Steve: You did your best. We're sorry that you sent so much and that you were trying to squeeze it through a link that couldn't take it. But after all, I mean, if there was that link that couldn't take it, you wouldn't have been able to send that much in the first place, if it was over a switched circuit. The problem was, you would know, the sender would know what the speed was that the router was able to, like the slowest link in the chain, the sender would know not to send too much.

So in this system, the system simply dropped packets. It doesn't even notify anybody.

And that was another conceptual breakthrough because some of the original designers said, wait a minute. If we're going to drop a packet, we need to send a message back saying we just, you know, sorry about that. And the smart people said, no, no, no, that'll create more problems because, if we've got a network that is congested, and so that we're having to drop packets, the last thing we can afford to do is generate more packets to send notifications back. The system has to work blind. And that made some people's heads explode because...

Leo: Engineers, you know, are going for 100 percent, perfection. And this is kind of accepting imperfection.

Steve: It was designing imperfection.

Leo: Yeah.

Steve: I mean, and this is - what I'm describing is the way, is what we have now. And you and I are talking to each other over it, Leo.

Leo: Yeah, it works okay. Good enough is good enough.

Steve: And so what's incredible is that, based on that foundation, based on the concept of taking what used to be a continuous connection, a continuous, reliable connection that worked at a known speed, they were able to just heterogenize this thing into chopping the same data up into pieces and launching it off and hoping for the best, literally hoping for the best. And that packet would travel through these links that interconnected routers, and maybe it would get to the other side. We are not sure. And we do know that, if a router's interface, its outgoing buffer was overfull, the router had formal, sanctioned permission to just drop it. Just let it go.

So this concept of best effort was a massive breakthrough. Clearly it had to be part of the solution of packet routing. And the question then was, okay, how do we get guaranteed connections, I mean, because what if we have to send a file? We just can't have bits and pieces of the file missing. That's no good. And if we're not going to get an acknowledgment, if we're not going to get, like, from the person who dropped the file bit as it was moving by, if we're not going to get them to tell us something, then we need some means for getting the recipient to notice that, wait a minute, I've got some missing pieces here.

And so that we will cover when we talk in detail about the so-called TCP, the Transmission Control Protocol, which was built on top of this seemingly rickety framework. And it works. We know it works. We use it when we get web pages, and we don't have bits and pieces missing from web pages. But the third thing that enabled a reliable protocol to be built on top of an unreliable protocol, this notion of even building something on top of something else, it was this notion of a hierarchy. And I would state that that's the third piece of brilliance.

We have packet routing, the concept of breaking up our communications into individual pieces and hoping for the best, and thus "best effort" is all we need by the nature of a heterogeneous network of interlinked routers with links that are coming up and down,

people, you know, routers may just be taken out of service and might not be there, we might be upgrading it, who knows what's going on? Somehow the data needs to get through the best it can. And best effort is all we can provide.

And so the designers said, if we're going to go packet routing, all we can do is hope. And but then they created a protocol hierarchy which is the third piece of genius, where you inherently have a carefully designed set of encapsulated protocols. For a non-engineer, it's a little tough to visualize. But the best analogy is we've all seen, for example, those nested dolls, the Russian dolls?

Leo: Babushka dolls.

Steve: Exactly. You've got a big one, you open it up in half, and there's another one inside. And you open that up, and there's another one inside. And you open that up, and there's another one inside that, and so forth. Well, that is exactly how the Internet protocols were built. Had they not been inspired, as they were, they could have just said, okay, well, we're going to have some sort of protocol, some agreement of, like, how to make this reliable and how to just simulate right off the bat a point-to-point connection the way we're used to having. I mean, they really liked this switched-circuit, point-to-point, leased-line style. That's the way all their hardware, that's the way everything, all their assumptions were based on that. So it was not easy to give that up.

But they said - so you could imagine that they could have immediately tried to recreate that from these random sort of loosely organized arriving packets. But they were smarter than that. They said, no, the base layer, the only thing we're going to do with the lowest level protocol - and again, the words I'm using are just habit because this is just the way we think now. But they weren't habit back then. There was no way to think about this. But so they said, all we're going to do to start with is address the packet to its destination. We're just going to say this blob of data wants to go there. And so that's all we're going to say. And we're not going to know if it got there or not. We have this crazy new network concept which is going to try to deliver it to its recipient. But maybe it'll get there or not, and we're not going to know. We're just going to create this thing, give it an address - oh, we're also going to tell it where it's from so the recipient could send something back to us. But nobody along the way is going to care. They're either going to send it to its target IP, its destination Internet Protocol address, or they're just going to drop it, the routers along the way.

And so that essentially is the IP protocol. The Internet Protocol, called IP, is this packet, also known as a datagram. And it contained four bits. I think that was interesting. I mean, they were - bits were expensive. Bandwidth was expensive. Everything was expensive. These routers were tens of thousands of dollars, not what we have now, things in the stationery store, for 49 bucks you could buy a router. I mean, this was heavy-duty equipment. And the bandwidth was still limited. I mean, even when they had links between routers, well, those were 64k. So it's not like somehow bandwidth was a gift of this system. If anything else, there was some overhead from all of this crazy packetizing and routing because now they had to have addresses. They didn't have to have addresses before because they were connected to who they were talking to. No more are they connected.

So there's overhead associated with this approach because you now have addresses for these bits and pieces. So, for example, the version number of the IP protocol, there were a number of bytes at the beginning of the so-called datagram. The least you can have is 20. So they only had 20 bytes that were available to deal with the mechanics of getting

this packet somewhere else. So they didn't want to waste bits. Consequently the version number is just four bits, four-bit version number. And zero, one, two, and three were used up before the Internet really happened. And so v4 is what finally did happen. And we've been using v4, most of us are still using it today.

And as we know, IPv6, v6 is coming along. And so in IP datagrams, almost all of them have, in those first four bits, they have a binary 4 in them. But increasingly, and in the future, there will be a 6 in there which will tell any equipment which receives it what the rest of the header looks like. And that's the other key. This IP datagram, this IP packet's version bits, they're the first four, and they have to be. So the very first four bits that arrive tell the equipment the format of the rest of the packet. That is, is this an IPv4 packet, or is it IPv6 or IP something else? So the designers, they took these 20 bytes. We know that four bytes, 32 bits, is for the destination IP. And another four bytes is for the source IP, the IP address that it came from.

Now, at this level, there are no - there is no notion of a port. That's an abstraction which was added by something that this packet contains, that is to say, by a higher level protocol above the IP protocol. The IP protocol, this lowest level one, is - all it's designed to do is take its payload, whatever it is, and it could be anything, in fact, that's one of the reasons, for example, we've talked about tunneling IPv6 over IPv4. This is the power of the genius of these original designers is they said, we don't know what our payload is going to be. We don't care. But so we're going to be very careful about not putting stuff in here that we ever have cause to regret later.

And I would say they did a perfect job. A couple fields have been redefined over time, but only sort of just shifting meanings of them, and not very much. The system as a whole has been just incredibly robust. And so, for example, if you're tunneling IPv4 or IPv6 over IPv4, what that means is that this ubiquitous universal IPv4 packet whose first four bits have a binary 4 in them, and all it contains then is IPv4 IP addresses, and some other housekeeping information, like a checksum and an indication of a total length of the whole packet, and also the length of the header. There's some additional stuff, but not very much. It's really a lean technology and protocol. That packet can contain anything else. Even a different IP version could be contained, IPv6, for example, contained in an IPv4 packet.

So that gives you a sense for, because these guys were so careful with the way this system was designed, with the way they thought this out, we've ended up with a set of technologies which for decades has survived virtually unchanged. And those three concepts that were going to drop this concept of a continuous connection in favor of chopping that same data up into pieces, adding some addressing information so that it can get where it's supposed to go, but then just turning it loose and hoping for the best, where we've got a bunch of well-intentioned routers that are all interconnected in a huge mesh, and each router will just do the best job it can to forward the data out of the interface which best takes this packet of data toward where it's trying to go. And so the router needs a routing table that we'll be talking about in the future to help it determine where it should be sent. And then this concept of, well, if the router can't do it, it just drops it and says nothing, doesn't even indicate that it dropped the data because that would create more problems if dropping data created data. So because the only time you would really drop it is in a time of congestion, so you don't want to generate more at that point.

Then, finally, this notion of a very careful protocol hierarchy, this nested - it's a nested hierarchy where one protocol contains the next one up the chain, in the same way that the Russian dolls contain each other. And with each wrapping you're only providing the information you need to. So, for example, just to give you a sneak peek of where we're

going to go in a week soon, the UDP protocol, or the TCP protocol, which do bring for the first time the notion of a port abstraction, they have port numbers in their protocol, but no IP addresses because that's provided by the wrapping protocol, the IP protocol, that knows about IP addresses but not port numbers. And in another example of the beautiful economy of this, there's another protocol that we'll talk about, ICMP, which is people are...

Leo: Ping.

Steve: Exactly, is ping and traceroute and is sort of like a maintenance plumbing protocol. Well, ICMP is different from UDP and TCP and, similarly, is not port oriented. So it doesn't have source and destination ports, yet it's able to be carried by the IP protocol which knows about IP addresses. So by carefully designing these nested layers of protocols, we end up with a system which is efficient and lean, and also very flexible, where over decades it has survived and, because it was so carefully designed, it's been extensible into the future. And as we know, we've got a big change coming along with IPv6. And we'll certainly be talking about how that relates to IPv4 as part of our ongoing series on how the Internet works.

Leo: Very interesting stuff. And I love it that you synthesized this into the three kind of key concepts because, I mean, I know all of these little bits and pieces. I've heard them over and over again. But really it helps you understand the decisions that were made by the guys who designed this stuff, and why.

Steve: Oh, yes, and to free themselves from the shackles of, oh, I mean, that's a huge conceptual leap. And then to say, well, but here, I mean, but wait a minute, maybe the packet can't get there. It's like, yeah, maybe.

Leo: Yeah. And then, and so what?

Steve: Yeah, but they didn't scrap it because of that. They said, well, maybe we can make it work anyway.

Leo: But that's how you make a robust system; right?

Steve: Oh, yes.

Leo: It's really quite clever.

Steve: Yes. In fact, that's a very good point, Leo. The fact that they knew that by design some packets might not get through meant they had to design for that. So there are, because there are many other reasons packets might not get through, not just a buffer overrun in a well-meaning router, but the router could crash. The link could go down. I mean, all kinds of unanticipated things other than just a buffer overrun could cause the same effect. And the robustness that they built in by design ended up making the entire

system robust. That's just so cool.

Leo: "Elegant" is the word that comes to mind.

Steve: Yeah.

Leo: Steve Gibson is the man in charge at the GRC, the Gibson Research Corporation. You can follow him on the Twitter, @SGgrc. You could visit his site, GRC.com. If you want to leave questions for next week, because we'll do a feedback session next week, GRC.com/feedback. But also you'll find there the show itself, the audio, 16KB versions, too, for the bandwidth-impaired, transcripts, if you like to read along with Steve. And I think for this one that would be a good thing to have, if you're studying up. This is the kind of thing that really would make great study material for somebody in an engineering class in college. It's like a great study guide, and you find that at GRC.com, along with Steve's many freebie products, his pro bono work, and of course his bread and butter, SpinRite, the world's best hard drive maintenance utility. It's all there at GRC.com.

We do this show, when there's not World Cup action, at 11:00 a.m. Pacific, 2:00 p.m. Eastern. I was a little late today because I couldn't - I had to watch the semifinal of the FIFA World Cup, the women's World Cup. But 11:00 a.m. Pacific, 2:00 p.m. Eastern at live.twit.tv every Wednesday. And of course if you miss the show, don't worry because you can always catch the download. We have audio and video available, not only on Steve's site, but at TWiT.tv/sn. Sierra Nancy. That's not right. Sierra November.

Steve: Leo is becoming a ham.

Leo: I am a ham. I've been a ham all my life. But now I'm going to have a license, license to ham. Steve, thank you so much. And we'll see you next week on Security Now!.

Steve: Thanks, Leo.