SECURITY NOW!

Transcript of Episode #359

Coddling Our Buffers

Description: After catching up with a few items of security and privacy news, Steve and Leo return to the Internet's "Buffer Bloat" problem to share the new solution "CoDel" (pronounced "coddle") that has been developed by several of the Internet's original and leading technologists and designers.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-359.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-359-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. He's going to debunk that whole RSA SecurID broken thing and then talk about buffering, how we can fix buffer bloat. There's a technology, I kid you not, called CoDel. Stay tuned.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 359, recorded June 27th, 2012: Coddling Our Buffers.

It's time for Security Now!, ready to cover your privacy and security needs with this man right here, our Explainer in Chief, Mr. Steve Gibson of GRC.com. Steve, good to see you again. We're a little late today because we just finished our live coverage of Day 1 of the Google I/O Conference, the developers conference.

Steve Gibson: Yeah.

Leo: Do you go to developers conferences? Microsoft has them; Apple has them; Google has them.

Steve: Back in the day, I used to. I used to go up to Redmond and hang out with Google stuff, I mean with Microsoft back in the day.

Leo: Yeah, I've been to a few of those.

Steve: And of course every COMDEX that there was I used to do breakfast with Philippe Kahn. He would always invite me up. And I'd go to parties and see Steve and Bill from

Microsoft and hang out. And I did go famously to the RSA Security Conference a few years ago, where I ran across Stina Ehrensvrd of Yubico and found the YubiKey. But in general, my sense is I can look at the schedules of the conferences, I can pick up the keynotes, I can look at the PDFs, read the research at length - which is, for example, what I did for our podcast today.

Several months ago, when the term "buffer bloat" was in the news, we did a podcast about buffer bloat. We'll review that briefly today because I want to explain the operation of the solution, which some fabulous Internet architects and designers have designed. And it's like, okay, now we just need it deployed. And that's the next step. But essentially they've come up with a really, really nice solution, which is really interesting to have when we look at how hard it has been for the industry to have something. This issue of how to deal with routing buffers on the Internet is as old as packet-switching because it's a problem that arises from that. And it has fought every prior attempt to wrestle it to the ground until now. We have a solution.

And then the big news of the week, we don't really have that much news, but the big news that got unfortunately picked up and completely blown out of proportion is - and I heard you mentioning it yesterday or Monday, I guess it must have been yesterday, about the scientists so-called cracking the RSA SecurID tokens. Ars Technica picked up on it. I got tweeted by everybody.

Leo: I knew you'd cover it. Yup. Good.

Steve: Yeah. So we have that to talk about, and just random tidbits and things. I also have the not-yet-published next book by Mark Russinovich that we'll talk about, that I have just finished.

Leo: Good. I just got Daniel Suarez's new book.

Steve: Oh, cool.

Leo: Yeah. So we can compare notes. All right, Steve. What's the story with the RSA tokens? I mean, that's a big deal, if they've been cracked.

Steve: Okay, yeah. And they have not been.

Leo: Oh. But that's what the headlines said.

Steve: I know. I know. Even Ars Technica, that I respect a lot and who often gets things correct, said - their headline was "Scientists crack RSA SecurID 800 tokens, steal cryptographic keys," which is not at all true.

Leo: In under 15 minutes. No?

Steve: Yeah, okay. So here's the deal. Bruce Schneier's famous quote is fantastic. It applies here. He says: "Attacks only get better, they never get worse." Which reminds us that, over time, academic researchers chip away at security. And we've talked about that in the seven and a half years of the podcast, over and over and over. We highlight instances of that, things like, for example, the MD5 we were talking about recently, the Message Digest 5, the MD5 hash, where as researchers continued to look at it, they poked at it and began to find some soft spots.

Well, there are standards associated with public key technology, asymmetric and also symmetric key technology, which researchers have long known about. The annoying thing about the paper that will be shown in August of 2012 at a major upcoming crypto conference is essentially an additional improvement on the efficiency of already known existing soft spots. So this, for example, is another reason that it's now time for us to be moving to 2048-bit asymmetric public key technology, up from 1024, because the researchers are continuing to look at these problems and getting better at it.

Now, way back in '98, so quite a while ago, there was a very sharp researcher, looks like I'm probably going to mispronounce his name, I would say Bleichanbacher, came up with what was sort of commonly known as the "million probes" attack. The idea is - this is the idea, both for the asymmetric public key attack and the symmetric attack, both of which were addressed in this paper that will be shown in a month and a half. And that is, any crypto algorithm that is in common use, the normal public key technology and symmetric key, they normally encrypt blocks of a certain size. For example, when we say "AES encryption," that's a 128-bit block that has various key lengths - 128, 192, or 256-bit keys. But the block size is always 128.

Well, what that means is that, when we're encrypting data, it's going to be uncommon, or unlikely, it's actually one in 128, that the actual length of our data would exactly fall into a multiple of the block size. So the question is, what do we do with the last block? And the solution is, the cryptographers have come up with what's known as "padding," where we pad the data out to the size of the block. So it turns out that the smart academics have figured out ways to get current state-of-the-art crypto systems to leak a little bit of information. And we've talked about some ways that happens, the so-called "side channel" attacks.

For example, right now we understand the danger of a side channel, for example, if a crypto algorithm used a different amount of power, depending upon the key that it was encrypting with, or if it took a different amount of time, depending upon the key. Well, those things that are like power or time, that are not about the data going in and out, they allow clever cryptographers and potentially bad guys, which is really the problem we're worried about, to glean some information.

So that's what happened in this case is back in '98 this million probes padding attack was demonstrated where the padding used for public key crypto could be used in order to probe for the key. The way this happens is you take something which has been encrypted, and for example, if you give it back to this algorithm to decrypt, and you haven't altered it, it'll say, yeah, fine, here, I decrypted it for you. But if you deliberately damage the encrypted data that you give it back to decrypt, once it believes it's decrypted, these algorithms check to make sure that the padding is correct, sort of as a form of checksum. And they will then respond, whoops, there seems to have been something wrong with what you just gave me.

Well, that bit of feedback turns out to be very, very weak, that is, the systems are so good that cryptographers have thought about this a lot and looked at it. And it's like, okay, a million probes. And then maybe there's a statistical chance that we can learn something, so forth and so on. Okay. Well, the efficiency of that was doubled in 2003 by a researcher, Kilma and his group, who were able to come up with an improvement. And then these guys, the authors of this paper and the subject of these news stories, made that attack essentially five times again more efficient.

So that today, with the actual technology in use, RSA SecurID Model 800 tokens, the Estonia government-wide secure ID card that has been issued, and a number of currently-in-use technologies in, as you said, Leo, in what has now become a matter of minutes, well, RSA got hit because theirs was the fastest to crack. I think it was 13 minutes. Now, we're not talking 13 lazy minutes. We're talking...

Leo: Hard-working minutes.

Steve: Hard-working, high-speed computer, so forth and so on minutes. Other of the systems are more like 90 minutes. But I saw some 96 hours to do some things. But still, I mean, we're no longer in millennia, which is where we would like to be with our crypto; but significantly, neither have the keys been stolen. All of this is one particular instance of a key that was used and encrypted can be determined, that is, all of this is about a hardware device - and we've talked about HSMs. Yubico has got one on the way, the Hardware Security Module.

The idea is, because we've given up trying to secure our computers, we just can't, we've decided, okay, we will move the sensitive stuff out of RAM into a little hardware box and plug it in by USB or Firewire or whatever. And we'll ship the stuff out to it, have it do the work, and then it just kind of gives us the answer. But so basically they're creating a black box. So that's been the solution we've come up with because it's the only way we've figured out we can secure anything because we've, as I said, given up on securing the machines themselves.

So what these attacks do is they would, for example, as we have talked about, if you're encrypting a file, you generate a pseudorandom key, and that's fast for bulk encryption. Then you encrypt the file with that. But then you encrypt just that key using public key technology, which is inherently slow. So you just encrypt the key that you used for encrypting the entire file. And that way you're able to minimize the amount of processing, but you have essentially the strength and flexibility of public key crypto without having to go so slowly. But this technology could allow them, in a distressingly short period of time, to figure out one key, that is, not crack the token, not get the sequence of numbers it's going to generate, not have it reveal its own secret private key, but just probe it and poke it to get sort of like one answer.

So what's most distressing is these problems have been known for a decade. In sort of following this trail back, I found the so-called PKCS #1 standard - PKCS is Public Key Cryptography Standard - on the RSA website. And what everybody is using now is PKCS #1 v1.5. And this is...

Leo: By the way, that's what the crack is against, not against particularly the SecurID token.

Steve: Correct.

Leo: It's just that it uses PKCS.

Steve: Yes. And everybody does. It's a standards-based API that gives you inter-token interchangeability.

Leo: The RSA folks were quick to point that out, saying that it's an academic exercise and not a useful attack. Does that seem fair, to characterize it that way?

Steve: Okay. One of my favorite cryptographers besides Bruce said - his name is Matthew Green. And I loved his quote about this. He said: "Never continue using a broken primitive only because the known attacks seem impractical today."

Leo: Right, right, right.

Steve: And so here's the point. PKCS v2.1 was published on June 14, 2002. Okay? We're at June 27, 2012. The 2.1, actually 2.0 even, fixed this problem. Nobody moved. Nobody adopted it. For 10 years, Leo, this thing has been solved. But it's like, well, yeah, we know there's problems with 1.5, but for the sake of backward compatibility, that's what we're going to support. And so, okay, a decade ago, maybe it made sense. Here we are 10 years later, and nobody has moved off of 1.5. These attacks are only effective against that standard, not the new one. And all of this has been known. But the industry has been sitting around, saying, well, exactly as you quoted RSA saying, there aren't any practical attacks; and, besides, we're selling lots of these. So we're not going to worry about it.

So that's the story. The good news is, this is, unfortunately, and we see this over and over and over, this is what is required to get people to get off the dime, to update themselves, to keep themselves current. It's not until something bad like this happens, well, it's exactly like Microsoft, that last year finally stopped using the MD5 hash, only last year. That was the weakness that has allowed Flame to get its foothold was that there was a newfound means of doing a chosen prefix attack on the MD5 hash. Had Microsoft stopped using it back when all the cryptographers said stop using it, we no longer can trust MD5, that wouldn't have happened. So we have exactly that scenario again. Ten years old is the spec that completely shuts this down. Nobody's using it. So the good news is they're probably going to start now.

Leo: And there's nothing to fear right now because it isn't in the wild exactly. Maybe it is now, I guess.

Steve: I would say, if you were - and that's the other thing is that the press ran around in a froth, trying to figure out what this meant. And it's, exactly as you said, a theoretical attack. I would say you don't want to hand your RSA SecurID token to bad guys and let them pound away on it for a few weeks. That would be an unwise thing to do. Although it's not like they can get anything from it. They have to have a specific target that they're using it to crack. So it does mean you want to keep your tokens close to you because this stuff does go from theoretical to real.

You can imagine, the NSA is probably drooling in some lab somewhere. It's like, oh, goody, now we have a way of hacking these things more quickly. And who knows what access they may have or tokens they may have acquired over time that they haven't been able to crack. Now this technology lets them do that. So this is the kind of thing that certainly has real-world application, yet it represents more of a policy failure throughout the entire crypto industry than it does anything else. It's just like, well, sure, the academics are poking away, but that's what they do. Now, this surprised people by bringing the attacks down into the mere hundreds of thousands from the millions, making it orders of magnitude faster. And suddenly it becomes real.

So anyway, that's what that was about. And this is, I mean, this is what we see over and over and over. This probably just needs to be regarded as the process. This is the process that the crypto industry has, sort of that tension that exists between the academics that designed these things and then tear them down, and the commercial side that wants to profit from them and is never in a big hurry to need all the old tokens obsoleted. They'd rather not have to replace them all. So every so often they have to.

Leo: I use Google Authenticator, which is a software-based program, does the same thing. And I'm trying to - maybe you would understand this. I don't understand it. But I'm trying to figure out what Google Authenticator uses. It says one-time passcodes are generated using open standards developed by the initiative for open authentication...

Steve: OAuth.

Leo: ...the HOTP, yeah, it's OAuth, using HMAC-based one-time password and time-based one-time password. But then it points to an RFC. So I can't tell if it's using...

Steve: Yeah, so this is very much the same thing that, for example, some of the YubiKey modes are. And it's the dongle that we talked about early on, the little football that PayPal and eBay were using, and I still have mine and use mine constantly. That technology is not the subject of this because this is more the so-called HSM, the Hardware Security Module. One thing was really interesting was that all these academics attacked relatively lightweight crypto solutions like these tokens, not the actual industry-strength hardware security modules. Why? Because they couldn't afford any of those. Those cost 10s and 20s of thousands of dollars. So they're much pricier, but they probably have the same problem.

But the good news is our little one-time password systems are - they're almost too dumb to succumb to this problem. You have to have a lot more of a crypto engine in there, somewhere where you're giving it work to do, and it's performing crypto functions in a black box mode and then telling you - and then giving you feedback. In fact, in one case there's one company's product is still in debug mode, where it dumps a log of what happened.

Leo: No. Oh.

Steve: [Laughing] I know. So it's like, thank you very much for the debugging log.

Leo: Hmm, that makes it a lot easier. Yeah, handy.

Steve: Made it much easier.

Leo: So it doesn't look like this Google Authenticator does use that particular technology, the PKCS.

Steve: No. All it's doing is it's either a counter or - all of these things, I've got the little VIP, the VeriSign identity system, in my...

Leo: That's not vulnerable, either.

Steve: No. Those just generate a nice little key. In order for it to be vulnerable, you have to have a device where, I mean, it's much more industry strength.

Leo: Got it.

Steve: You give it some work to do, it's got a crypto engine in it, and it actually performs the decryption for you and gives you a result. All of our little authentication technologies are just pseudorandom number generators. So again, not victims of this kind of problem.

Leo: Excellent. Good news.

Steve: I picked up an interesting tidbit that I actually forgot to add to my notes, but then I saw it again. I went, ooh, I just wanted to mention this, just as an indication of things to come which I'm glad for, and that is the news yesterday that the FTC, the U.S. Federal Trade Commission, has hit the Wyndham Hotels chain with a lawsuit over three hacker breaches in the past two years. And so the story reads, "Wyndham Hotel Group just took the fourth blow in a quadruple whammy: First, it was hit with three digital breaches over two years that affected more than half a million customers. Now, the Federal Trade Commission has filed a lawsuit against the firm for allegedly misrepresenting the security measures that ought to have prevented those hacker intrusions.

"In a press statement Tuesday, the FTC claimed Wyndham had subjected consumers' data to an 'unfair and deceptive' lack of protection that led to a series of breaches of Wyndham hotels and those of three subsidiaries. The statement describes a series of three attacks on the hotel chain and its franchisees, beginning in 2008, that first compromised 500,000 credit card numbers stored by the firm, followed by attacks that breached another 50,000 and 69,000 accounts at other locations.

"The commission claims that those breaches are the result of Wyndham's failure to properly use complex passwords, a network setup that didn't properly separate corporate and hotel systems, and 'improper software configuration' that led to sensitive payment card information being stored without encryption. The FTC contrasts that lack of protection with Wyndham's privacy policy statements that claim to 'recognize the importance of protecting the privacy of individual-specific (personally identifiable) information collected about guests, callers to our central reservation centers, visitors to our websites, and members participating in our Loyalty Programs,' and promise the use of strong encryption and firewalls."

So I saw that, and I thought, oh, good. I mean, unfortunately, it's this kind of publicity that again, as I've often said, will end up being the motivation in the boardrooms of the CEO to say to the CIO or CTO, okay, tell us this cannot happen to us. And they squirm around in their chair a little bit and say, oh, well, you know, we need more money or time or whatever. So it's just good news that we have some oversight like this. And it's weird that it's the Federal Trade Commission, but it's because Wyndham is not doing what they are saying that they're doing.

Leo: It's interesting. If they start suing people for this kind of stuff, that's very interesting.

Steve: Yes. I wanted to mention also that I had just a fantastic experience generating an SSL certificate last night. And you don't often hear someone saying that.

Leo: Never hear that.

Steve: No. And this is DigiCert once again. I talked about them before when I switched GRC, when I dropped VeriSign, now owned by Symantec, like a hot rock and switched over to DigiCert and was able to get a multidomain EV certificate. I went through the whole extended validation process once. GRC has the advantage of having been around for 20-plus years, so D&B knows about us and so forth. So it was easy to see we have established addresses and phone numbers and so forth.

But the certificate that I got had a couple extra slots in it because, unlike VeriSign, that was going to charge me something like $1,300 per, I was able to get a four-slot single certificate for $550 or something. So, I mean, radically more affordable. It's the thing that, I mean, I've wanted to have, of course, an EV cert for a long time. I ought to have one. And finally it was my switch to DigiCert that enabled that.

Well, as I'm moving more and more toward GRC being always HTTPS, I realized that the media server that we've got, where all of the archived podcasts, for example, are stored, as well as the videos that I serve on the site, the media server didn't have an SSL certificate. So last night I thought, huh, I wonder if I can actually make this happen. And so I went back to DigiCert, logged in. They found my account, and it said, oh, you have two free empty slots. And I said, oh, that is just too cool. So I went over to the server, the media server, had it generate a so-called certificate signing request, a CSR, dropped that onto the website, pressed the button. It took maybe about five minutes for them to verify everything, see that everything was current and correct.

And it was only because I was adding a subdomain to GRC that this was an automated process. So, for example, I have www.grc.com and GRC.com. Those were the two out of the four slots I had filled. Now I have media.grc.com. And in, like, five minutes I received email with the signed certificate that I dropped onto the server, and we now have SSL. So I know this sounds like a commercial. I'm getting of course nothing from them except I want to help them because they deserve it. These guys rock, and I am so glad that I made the switch to them. So that's DigiCert, and I just can't say enough good about them.

Leo: Good.

Steve: And while I'm saying good things, Leo, I've just finished our friend Mark Russinovich's latest book. His prior book was "Zero Day," which I read and liked and talked about. Now, the bad news is no one can get this yet. Mark said, when I finished yesterday and wrote to him and said, okay, this was really fun - and in fact, what I loved about this, I was thinking of our podcast listeners. And, I mean, I was depressed and chilled because I would recommend this when it's available, and it won't be until September, and I will remind people when it is. In fact, I'm going to see if we could get Mark to jump on and talk to us because...

Leo: Oh, that'd be great.

Steve: ...this is absolutely true. It's obviously a fictionalized account. But this talks about Stuxnet and Duqu, and it's an adventure written around the same two central characters, Jeff and Daryl, who he introduced in his first book, "Zero Day." So we continue following them. They're ex-NSA and CIA, but computer people, sort of who should not be out in the field, but they end up going out anyway.

But as I'm reading this, I'm thinking, okay, anybody who we want to explain how bad, unfortunately bad things are with security, this is the book because it's very accessible, it's very readable, and Mark just lays out how really sad the current state of cyberwarfare is, I mean, the fact that the U.S. electrical grid may well already be infiltrated with malware. I mean, here we're chuckling about apparently the U.S. in Iran with the Olympic Games project, having designed Stuxnet and managed to get it into the nuclear enrichment facilities in Iran. And just last week or the week before I was saying, yes, well, the joke may be on us because we're getting our chips from China that's in all of our networking gear. So we hope that those don't have any extra functions added to them.

Anyway, this book is "Trojan Horse," not available for a few months, but I will let our listeners know when it is and see if we can have Mark come on to talk to us about it because, again, it's a great book. But oh, my goodness. I think our listeners will find it fascinating as a fictionalized account by somebody who really knows this stuff, and an incredibly accurate portrayal of how sad things are right now. I mean, you really just want to unplug. It's scary.

Leo: Mark Russinovich was originally with a company called Sysinternals. Microsoft acquired them.

Steve: Founded.

Leo: He's a programmer. Founded it, yeah.

Steve: Founded, he and Bryce founded Sysinternals, which generated some of the best Windows utilities that have ever been created. And now he's a technical fellow at Microsoft, which is their most revered slot. And it turns out he can write, too.

Leo: He's a technical fellow.

Steve: I promised a second sort of surprising SpinRite story last week about SpinRite's success in recovering solid-state media, which we're beginning to see more and more because I've mentioned it. And so this is Bill Murray in Wilmington, North Carolina, said "Hi, Steve. I'd like to share another SpinRite testimonial with you. I recently took a 7,000-mile motorcycle trip." How do you ride 7,000 miles?

Leo: With a sore butt.

Steve: Without running into some ocean somewhere.

Leo: Oh, yeah. But he went across the country and back, I guess.

Steve: He went around in circles. Yeah, says, "I recently took a 7,000-mile motorcycle trip from North Carolina," where he's based, "to visit Colorado, Utah, and Arizona." Okay, so he kind of made a big loop. "During the trip I was taking several hundred photos per day. Some were JPEGs taken with a point-and-shoot camera while on the move, and others were RAW plus JPEGs from my Canon DSLR. Each evening I was copying my camera cards to both my Netbook, equipped with a 96GB SSD drive, and to a 32GB USB thumb drive, since I was quickly filling up my camera cards. Unfortunately, I soon ran out of space on both the Netbook and my largest USB thumb drive.

"To conserve space, since the SSD was almost full, I began using only the USB thumb drive. After returning home, when I went to import the USB thumb drive photos into Adobe Lightroom, I discovered that three folders, three days' worth of photos, were corrupt and unrecoverable. Remembering that you and Leo had previously discussed a testimonial during a previous Security Now! podcast where someone used SpinRite to recover some Flash media, I broke out my copy of SpinRite which I had previously purchased to pretest drives for my Windows home server.

"I proceeded to check the USB thumb drive. After completing the scan, to my amazement, I found it had recovered my photos in the corrupted folders. I had some recovery messages about not being able to relocate some files, since they were marked "system" and hidden. But it had worked. Like many others, I want to thank you for providing this outstanding product. Also, a huge thank you for the time and effort you put into the weekly Security Now! podcasts with Leo Laporte. Bill Murray, Wilmington, North Carolina." Thank you, Bill.

Leo: Thank you, Bill. Yeah. That's nice.

Steve: Yeah. So I've been sort of, for a while, thinking, well, I wonder whether SpinRite's life is coming to an end as our media stops spinning. But although the name will be a little antiquated...

Leo: Call it FlashRite.

Steve: It sure does, well, and I actually understand why. Because all of this technology, they have controllers, and they have error correction. And that's SpinRite's forte is dealing with that problem with error-prone media. And in the same way that we have a problem with magnetic storage because basically they're always pushing the limit, trying to cram as much possible data into as small a space as possible, always making it just reliable enough. Similarly, they're putting, for example, in the case of MLC, or Multi Level Cell storage, they're putting analog voltages into individual capacitors in these memories. And as those voltages drift, the bits they read back are not going to be the same, so they need error correction in order to recover from a known expected error rate. And that's where SpinRite lives is dealing with those kinds of problems. So it does look like we've got plenty of future left still.

Leo: Good. I'd hate for you to starve.

Steve: And remember, people have been using it for 20 years, and it's still going strong. So it does represent an investment for the future.

Leo: There you go. Steve, what is CoDel?

Steve: Okay. So CoDel is the way you pronounce the name of what will end up probably being the buffer management strategy which ends up dominating the Internet over time. Sadly, we all have devices which are probably using what's known as "tail drop," which is the official term for if the buffer's full, and there's no room to put a packet, well, what can we do? Throw it away.

So let's rewind a little bit and remember how we got here. The fundamental, always understood problem with an autonomous packet-routing network is the need for buffering because, if you imagine like a lattice grid of nodes all connected to each other in a very complex topology where customers are located off of different of these nodes, and they put their Internet traffic onto a given node, that is to say, a router, and it sends it to the next one, which sends it to the next one, which sends it to the next one.

So you've got all this traffic moving, jumping from one router to the next as it goes from its source to its destination and back again. And if you do like a traceroute command, the number of hops, as they're called, between routers, certainly it varies. But we're used to seeing something like 13 to 17, something like that. And we've talked about how the so-called "Internet diameter," that is, diameter of a circle is of course the two points that are the farthest possible away. Similarly, the diameter of the Internet would be the two locations that had the largest number of routers between them so that the data has to go through that many hops. And there was a problem that the so-called TTL, the Time To Live, was originally set at a number that was at the time reasonable, maybe 16 or 32. Generally programmers like powers of two.

But it turns out that, as the Internet grew in size, more routers were added, and it was possible for an operating system to generate a packet that would never be able to make it to its destination because its time to live would expire before then. So because we've got all of this, sort of this ebb and flow happening statistically, but without any overall plan, just sort of every packet for himself, it's necessary to buffer both packets coming into a router because they might be coming in from - you might have five or six all arrive at once from different interfaces. And then the router figures out which interfaces each one needs to go back out of. And you then need to put those in a queue, that is to say a buffer, so that they can move as soon as there's space available, and time, on the outgoing wire.

So buffering has always been present. The problem, as we discussed it when we talked about buffer bloat in detail, is that there's been a tendency, as the price of memory has fallen, now there's just more and more RAM in these single-chip processors that all of our consumer routers are based on, and it's counterintuitive to a network engineer to think, wait a minute, I have a perfectly good packet here, and I have free buffer space. Why should I drop it? I've got room for it.

And the reason is there's a very subtle interplay between the overall efficiency of our protocols and time delay. Big buffers create a time delay where more data is coming in than is able to go out. I mean, anyone can sort of think of the model of a funnel, where if you put a lot of stuff in the wide open end of a funnel, and it's going out slowly, well, the funnel itself captures the material that's trying to get out through the smaller opening, and it holds it for a while. So any given grain of sand sits in this funnel queue, waiting for its chance to finally get out of the funnel. Rather than, if you were only pouring in sand at the rate it could flow out, then the delay through the funnel would be as near to zero as it can be, just the grain of sand's own passage through. There wouldn't be a backlog.

So it's the backlog which represents a problem, and backlogs form any time you have a transition from a high-bandwidth connection to a lower bandwidth connection. If you are sending things in at the high-bandwidth speed, they're going to have to back up. They're going to backlog.

So many, as we've discussed before, many strategies have been designed. In fact, it's an acronym soup of strategies. I mentioned tail drop, which is the dumb one, the dumbest one, which is buffer everything until when a new packet comes in that needs to wait, and there is no place for it, we just drop it. Now, again, as we've talked about packet switching, the way that works, it's inherently in the design of the Internet to tolerate dropped packets. Routers are supposed to drop packets. And what's interesting is that's informational; that actually the dropping of a packet in transit indirectly sends information back to the sender because the recipient never receives the dropped packet, thus never acknowledges the receipt.

And so the original protocol designers designed the Internet to function with the assumption of dropped packets and to deal with it. So engineers that engineered larger buffers, and unfortunately these things chain, so if every router between here and there has a big buffer, then you can end up with many, many seconds required to get your data to transit from one place to the other. And if we were only transferring big files, that wouldn't be such a problem. But you and I, Leo, are having a conversation in near real-time over the Internet.

Leo: It's kind of amazing.

Steve: And so we don't want to have a grain of sand stuck in a funnel where that's my voice trying to get to you. And so these deep buffers really do not help anybody. What we want is we want them to catch bursts. We want them to, like, be a surprise. Oh, suddenly more came in than we were expecting. Well, as long as it doesn't persist, as long as that was a surprise event that will then quickly be removed, that's fine. That's proper use of a buffer. But not something that just sits there full and clogged with sand running over the sides of the funnel because more is coming in all the time than is able to get out the other end.

So the most popular solution is something called "RED," which is the acronym for Random Early Detection. It's also been known as Random Early Discard, or Random Early Drop. The idea there is we don't wait to have no room because obviously we have no room. There's nowhere to put an incoming packet. Instead, as we begin filling up, we increase the probability of discarding packets even though we've got room. And counterintuitive as that is, you can imagine that that inherently causes the buffer to resist getting full. It's discarding things before it's actually full because this is an acknowledgment of the truth, that discarded packets send a message to their sender. Protocols are designed so that somebody that's sending things and stops getting them acknowledged will go, oh, maybe I ought to slow down. And it's like, yes, that's the message we're trying to send you, slow down.

And so all these systems tend to throttle. The problem is that, despite an amazing amount of industry being focused on this problem, as I said, acronym soup, all kinds of solutions proposed, there has never been one that works across the board, is independent of bandwidth, is independent of roundtrip time, is independent of traffic type. Many of these things can be tuned for specific situations. If you knew that your bandwidth was such and such, if you knew that your typical roundtrip time was going to be something, if you knew whether you had bursty traffic or more slow buffer, or if you knew you had UDP stuff or TCP, if you put all kinds of constraints on, then anybody could carefully tune a solution for a given set of traffic.

The problem is we no longer have homogeneous traffic. We have heterogeneous traffic. We've got all kinds of bizarre stuff happening, peaks and valleys, widely ranging bandwidth. And in fact, what's interesting is that the problem with residential buffer bloat is largely caused by WiFi. And I think it was - I was watching Tom and I think it was Brian Brushwood, they were talking about WiFi and maybe, like, grumbling. I think he was using some sort of non-WiFi wireless to do some interacting with your studios yesterday. And he went off on a rant about how useless free WiFi is in coffee shops or anything because it just...

Leo: It's horrible.

Steve: Yeah, it turns out that the reason that is, is that what we can't see is that, as you move your hands around your laptop, as you rotate, as people walk by, the instantaneous bandwidth between you and the hotspot is jumping up and down and changing, and it's doing that separately for all the people who are trying to use it at the same time. I mean, it's amazing it works at all. And so as a consequence it really doesn't work very well.

So the challenge is huge. So the news is, and this is just amazing when you put it in the full context of how big a problem this has been, how intractable it has been, how the best minds in the industry have been focusing on this, in fact, even Van Jacobson, who is the co-designer of this solution, he gave a speech six years ago, in 2006, and wrote a paper that never got published that was beginning to sniff at this, beginning to talk about this solution. But it just never got adopted.

Now, the other problem is one of scale because we have big iron routers, like in Level 3 and at AT&T, with high-speed fiber optics, where the actual routing is done in silicon, and we have low-end blue-box pieces of junk that cost $30, that run off of a wall wart power supply, that have a low-power chip barely running Linux in a consumer router. And the problem is the same. It's at a different scale, but fundamentally nothing changes. And so what they were looking for was something that solved this entire problem.

So this was developed by Kathleen Nichols and Van Jacobson. Van Jacobson is known for having come up with a next-generation flow control for TCP. It's called "Van Jacobson's algorithm." And he is regarded as the guy who singlehandedly saved the Internet from collapse in the late 1980s and early 1990s, that is, the first version of TCP we're no longer using, and it would not have worked. We needed a better means of flow control, and it was Van Jacobson who came up with the algorithms everybody is using today, Tahoe and Reno. You've probably heard those acronyms.

Leo: I've been there, but no.

Steve: Those are some of the approaches used for congestion avoidance in TCP. So what Kathleen and Van Jacobson figured out how to do was they developed an algorithm, which I'm going to describe to you, and it's tricky. But one of the things that's so nice is that it absolutely fits the need. That is, it can run on, and is actually running on, Linux-based home routers. The CeroWrt does have CoDel, the CoDel controlled delay buffer management in it now as an experimental platform.

So CoDel is parameterless. There's no knobs, as they put it, for operators, users, or implementers. There's nothing that needs to be adjusted. The same algorithm can run in a $30 piece of plastic as runs on huge big iron, where all the routing is being done in silicon at the high end. Same algorithm. Unlike many of the algorithms that have been proposed, as they phrase it, it treats good queue and bad queue differently. And by that I mean it keeps the delays low while permitting bursts of traffic. So it's exactly as I said we want.

I need to make sure I use the right words and that I define these terms because a buffer is like the static container for the data. The queue is the actual lineup of the packets. So I'm going to try to make sure I use the right terminology here. So the queue is the present list of waiting packets that are lined up in a queue in the buffer, waiting to be sent. CoDel adapts to dynamically changing link rates with no impact on utilization. And that's one of its coolest factors. There is a PDF which was published by the ACM.

I'm trying to think, if you Google "controlling queue delay," Leo, I think it's the first link that comes up, controlling queue delay. And same for our listeners, of course, if anyone is interested. There's a PDF. And Figure 7 on I think it's page 10, looks like page 9 of the PDF, shows how this performs relative to the random early detection and the tail drop algorithms in the face of changing bandwidths. And it's just spectacular what they have done. And as important as anything else, it is simple and efficient. That is, because it needs to run in an underpowered chip, or be easily implementable in router silicon, it's very important that it's not overly complex.

So what is different from this algorithm from all prior approaches, the general acronym is AQM, Active Queue Management, and that's the term that covers any approach of any sort for trying to intercede in managing the queue, managing the list of waiting packets that are sitting in a buffer such that you are able to respond to bursty traffic, where at the same time you discard and drop packets intelligently to send the messages to the existing algorithms that they need to throttle themselves.

So what's extremely cool about this is that they maintain what they call a "single-state variable," which is the length of time that a packet has been in the queue, that is, what they call the minimum queue length, but it's measured in milliseconds. And having more resolution ends up being good for this, and that's generally easy to do these days. They use the term "packet sojourn time," as the packets sojourn through the buffer. And I'll just abbreviate that PST so I'm not having to say it all the time. But so think of PST, this packet sojourn time, as the length of time a packet sits in the buffer while it's in the queue moving forward as packets are sent out.

So their algorithm is, and this is the result of extensive testing in all kinds of networking configurations, they have what they call their "target acceptable PST," that is, what they call the "standing queue duration," is five milliseconds. That's the number, five milliseconds. So this CoDel algorithm has as its target that no packet will be in the queue longer than five milliseconds, that is, within a - it's a little bit more flexible than that - within a maximum of 100 milliseconds, so one tenth of a second. The queue has to have dropped to five milliseconds sometime within a hundred-millisecond window. So this so-called PST is a minimum queue length that has to have occurred within a tenth of a second.

When that PST, that packet sojourn time, has exceeded five milliseconds for at least 100 milliseconds, then a packet is dropped. So even when the buffer's not full, but if the length of time a packet has been in the buffer has not dropped down to their target of five milliseconds any time within a hundred millisecond window, then they start discarding. Essentially, this just sort of switches into a discarding mode, and they drop the first packet.

The next drop time, and this gets a little hairy, but I'll explain what this means in a second, the next drop time is decreased in inverse proportion to the square root of the number of drops since the dropping state was entered.

Leo: So the more drops - go ahead. You lost me.

Steve: The next drop time is decreased. The drop time is decreased, so that means the drop rate is increased, in inverse proportion to the square root of the number of drops since it began dropping. What that means in practice is it accelerates.

Leo: Faster, okay.

Steve: Yes, it gets faster until it gets back to its target of five milliseconds. And believe it or not, that's it. That's all there is to it. What this ends up doing is CoDel then acts, in engineering terms, as what's called a "closed-loop servo system" that gradually increases the frequency of dropping until the queue is controlled. And it turns out in packet-switching theory there's actually a well-known relationship of throughput to the probability of dropping, so that dropping ends up helping your throughput. And they end up then, they have on their site, and there's an appendix referred to in this PDF where they post the pseudocode, because this is all public domain, obviously this will be an RFC. And I imagine, I hope that router vendors run to implement this, that the big router vendors like Cisco and so forth offer firmware updates, and even little routers, Linksys and the Cisco SOHO routers offer firmware to implement this because it turns out it is simple to do.

Even this inverse square root drop probability, it turns out that the inverse square root can be computed efficiently using only integer multiplication. So even that isn't going to cause any great problem. And the entire algorithm is implemented with just four variables: the time of the first drop, the time to drop the next packet, the count of the numbers dropped since going into the dropped state, and then just a Boolean that says whether we are currently dropping or not. And so it is really simple to do this. I mean, it's no more complex than statistically dropping at a probability based on how long the queue is, which is the RED, the Random Early Drop or Detection approach. It is simple to do.

In their paper they demonstrate exactly this WiFi nightmare of rapidly changing bandwidths. And, for example, they start off at time zero with a 100Mb channel. Then they drop it to one tenth of that, to 10Mb; then again by a factor of ten to 1Mb; and then jump it up to 50; and then up to 1Mb; and then down to 10 again. And they show for all of those changes what CoDel does versus what the standard buffer management of either random early detection or tail drop does. And it's just shocking how well CoDel performs. It very quickly adapts. It responds to the change. And this little closed-loop servo that I described drops, I mean, intelligently starts dropping to keep the overall delay through the buffer short, yet also signaling in an optimal fashion to any TCP traffic that is going through that it needs to slow itself down. And the consequence is essentially, after decades of trying, we've got this problem solved.

Leo: Yeah, except you can only get it in one router.

Steve: Yeah. Nobody has it yet.

Leo: And I doubt this is the kind of thing you can do in firmware. Maybe, could you do it in firmware and update existing...

Steve: Oh absolutely.

Leo: You could. So it's just software.

Steve: Absolutely.

Leo: Just an algorithm.

Steve: Right now, yeah, right now firmware is maintaining, probably in a brain-dead fashion, just having a buffer, and is probably huge. It's probably...

Leo: So the fact that they've got these giant buffers, because they have so much RAM, is controlled by firmware, so you can change how this works.

Steve: Exactly.

Leo: Even if you have all the hardware, you don't have to use it.

Steve: And that's been the problem, is that our own, our SOHO buffers in our SOHO routers, they now have buffers that are multiple seconds long in terms of our upstream bandwidth. So when we're uploading something or trying to play, like, a game while somebody else is doing something in terms of bulk bandwidth transit, suddenly all of your interactivity, all of the game interactivity just disappears because, even though it's well meaning, this SOHO buffer fills up, and it's several seconds of delay.

Well, the bulk transit, the data being moved doesn't care. But somebody who's trying to do an interactive online game, it just comes to a stop because, I mean, several seconds, think about that. I mean, you and I could not be having a conversation. We've all seen newscasters trying to talk to somebody via a satellite delay. And you have to have some skill in order to control yourself and wait for that satellite delay and manage that kind of delay. It's really difficult when you want to be interactive and operating in real-time. So what they've got is something with minimal processing, minimal state. You know that one of the things that they're very proud of is that they only need to track the minimum delay through the buffer. They do not need the average. And that's significant...

Leo: That's a big difference, yeah.

Steve: ...because, yes, in terms of computation, to do the average you need to add up all the delays and divide by the number of packets. To do the minimum, all you need to do is see what the timestamp of the packet as you're removing it is, and that's how long that guy's been there. And then you maintain the smallest one you've seen within a window of a hundred milliseconds. And if that's never down at your target of five milliseconds any time during a tenth of a second, then you kick into beginning to discard, and you increase the rate that you're discarding until you bring it down. So it's got some state in it. It kind of switches into we need to start signaling to the people who are using this buffer that they need to back off, and it does it in a way that seamlessly fits with the Internet's protocols.

So having this is the first step. Getting it ratified, getting the word out, and getting it then moved into firmware gives us some hope that we're going to get this buffer bloat problem solved. But users can keep an eye on firmware for their own routers because remember the problem we have is where we go from high bandwidth to low bandwidth. That's the choke point, that funnel problem. And all home networks have 100Mb local networks suddenly squeezing into a 2Mb uplink through a DSL or cable modem. That's where their buffers fill up. And so that's where we see this problem. And what would happen is this would then provide signaling back to the machines in the network to slow down in order to allow the residential buffer to stay real-time, yet actually increasing the overall throughput, counterintuitive as that is. They show charts where they show the throughput under these varying conditions, even though they're discarding intelligently, and it's right up between 90 and 100 percent; whereas the traditional algorithms collapse down into the low 10s of percent of overall throughput. It's just - it's fantastic.

Leo: Steve Gibson explains all. And now if we can only get the router guys to put it in, we'll be good. If you want to follow along with the transcripts - couple of people have said in the chatroom "I'm going to read this one" - you can go to the website, GRC.com. That's where Steve lives. He puts transcripts and 16Kb versions of the show, audio versions of the show there every week. You can also get full video and audio versions at our site, TWiT.tv/sn. Steve will be answering questions next week; right?

Steve: Well, will I?

Leo: Oh, it's the Fourth of July.

Steve: Yeah.

Leo: I don't know.

Steve: Well, the memo I got said that you guys...

Leo: Take the day off?

Steve: Well, that we're not going to be prerecording.

Leo: That's a shock.

Steve: I know.

Leo: Do you want to do a prerecording? I can check.

Steve: Of course.

Leo: I can check. I'll check with the people. I'll check with the people. But you're right, it is the Fourth of July next Wednesday; isn't it.

Steve: It's the Fourth of July. I had assumed we would find some other time to record, but I got a note saying nope, there will be no prerecording of podcasts. It's like, oh.

Leo: Lisa's trying to protect me.

Steve: My thoughts exactly. It did come from Lisa.

Leo: Yeah, she's very aggressively protecting my time. But she doesn't understand that this show must go on. We don't miss episodes of this show.

Steve: We never have.

Leo: Never have. Let me check, and I'll get back to you. But meanwhile, if you have a question, if it's this week or next, go to GRC.com/feedback and fill out the feedback form.

Steve: Yes.

Leo: We will do a Q&A episode in our next episode, which is unknown when that will be. And of course there's lots of other good stuff there, not just SpinRite, the world's finest hard drive maintenance and recovery utility. You must have it. But also...

Steve: It works.

Leo: It works, even on SSDs. But also lots of other great stuff that Steve gives away for free because he's just a nice person. GRC, Gibson Research Corporation, GRC.com. You can follow Steve on Twitter, @SGgrc. He's also got some other - he's got a "vlc" for the "very low carb," @SGvlc. He's got @SGpad for the pads. I'm hoping we're going to get one of these new Nexus tablets in here, and we can take a look at that.

Steve: The Nexus 7, yes.

Leo: Yeah, it looks pretty sweet. We'll have details on that. Thank you, Steve. And you know what, folks, go to the calendar at TWiT.tv, and it will have - whenever the next recording is will be on the calendar.

Steve: And anybody who's following me on Twitter. When I know what the story is...

Leo: Oh, you'll tweet it, of course.

Steve: ...I will tweet it and let people know.

Leo: Very good. Thank you, Steve. Steve Gibson.

Steve: Thanks, Leo.

Leo: Have a great day. We'll be back next time, unknown when, but next time on Security Now!.

Steve: Bye.