SECURITY NOW!

Transcript of Episode #164

Sockstress

Description: Steve and Leo discuss a class of newly disclosed vulnerabilities reported to exist in many operating systems' implementations of the fundamental TCP protocol. Two security researchers, claiming that they could not get anyone's attention, disclosed far too much information in a recent audio interview - leaving little to the imagination - and exposing the Internet to a new class of DoS attacks. They'll certainly get attention now.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-164.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-164-lq.mp3

INTRO: Netcasts you love, from people you trust. This is TWiT.

Leo Laporte: Bandwidth for Security Now! is provided by AOL Radio at AOL.com/podcasting.

This is Security Now! with Steve Gibson, Episode 164 for October 2, 2008: Sockstress. This show is brought to you by listeners like you and your contributions. We couldn't do it without you. Thanks so much.

It's time for Security Now!. This is the show which helps you, we hope, understand how security works and protect yourself online. Here he is, king of security, Steve Gibson, from his fortress...

Steve Gibson: Sometimes it may confuse you, but we try to do our best not to have that be the result.

Leo: I think that when people listen, and they don't understand, the universal reaction - and I include myself in this, Steve - is that it's our fault, not yours. You do a very good job of explaining it. And these are just difficult subjects, and sometimes so difficult that you just, you know, you can't understand it.

Steve: The only thing I would wish for, and this is just not practical, would be like a rehearsal of the whole thing. There are some topics...

Leo: We can rehearse. You want to rehearse?

Steve: No. No.

Leo: It's too much work.

Steve: That's my point is that, I mean, there are some things that I've tackled where I have afterwards been unhappy with my own presentation, thinking okay, you know, now that I've done it once...

Leo: I think that's true. I think that's true. Sometimes - and I blame myself in that case because I think I could probably say, well, let me go through this and make sure I understand it. I'll do that more also. But Steve, a standing invitation. Anytime you want to - you say hey, I think I could do that better - I'm here. I'm here all day, all night. Sometimes it seems that way. So just give me a holler, and we'll do it again. Because I - these are sometimes complicated. Very complicated. Now, today we have a kind of a shocker.

Steve: We have a shocker. We're not doing a Q&A today. Something happened just today, and all of our listeners are going to be writing to us asking, if they're not writing right now, asking oh my god, have you heard about this, what does it mean, blah blah blah. The other reason I want to talk about it is there's a huge amount of confusion that has already surfaced. Basically a couple of security researchers are claiming, and I'm believing them, that they have found some serious problems in, they're saying, every TCP stack that they have looked at; and that they are able to bring sites down with essentially a low-bandwidth denial of service. And if you put "sockstress" into Google, you'll find pages. This thing just happened today as a consequence of an audio interview that they gave.

What I'm really unhappy about is that they let the cat out of the bag. They did not handle this in the way Dan Kaminsky did. As I'm listening to this MP3, I quickly understood what it was. And I was thinking to myself, okay, stop talking. Stop talking right now. Don't say anything more. But they kept talking. And...

Leo: Well, this is what you get from a guy named CrackersChild. I mean, clearly the guy's not a white hat.

Steve: Well, they're - we'll talk about this during the podcast because - and I'm going to explain to people what it is that they've done because the cat's out of the bag. Anybody who was a packetsmith, who is the kind of person who could implement this, all they have to do is listen to this audio, which is now spread all over the web, and they will...

Leo: That's terrible. That's very irresponsible.

Steve: It's really irresponsible. I mean, they...

Leo: So you're saying the minute they discovered it they just said, hey, here's what we found, instead of going to the vendors and saying maybe work, as Dan Kaminsky did, maybe work with them?

Steve: They say they've been playing with this since 2005.

Leo: Oh.

Steve: They say that they are in discussions with vendors but that they haven't been going very far or very well or something. And then they gave one presentation sometime before. They're giving another one on the 17th and 18th in Helsinki, and where they're going to demonstrate this. But already in this MP3, which has been linked from Slashdot, anyone listening to it who could implement this will know how to. I mean, I could do this in about an hour. I have no interest in bringing sites down. I never have. But there's, I mean, this is easier to do than what Kaminsky found. And it's more obvious how to do it from what these guys have already said.

So where Dan really did keep a lid on this for three months until the patches were out, these guys have not. And what they've said means that by the end of the day attacks will be developed. I mean, it's that easy to do this. And they're saying they can bring any Windows or Linux or BSD box and routers down. And based on what they've said, I believe them. And anyway, that's the topic of today's show: Sockstress.

And I'm going to, because this is not rocket science, and because they've essentially gotten now a published MP3 audio which anyone who has the ability to degenerate packets and understand TCP, this is all anyone needs. This is much - they've given the world much more than Kaminsky did. And remember, it took a week there between the time he told us there was a problem he was going to - I guess it was two weeks because it was going to be a month between the time he let there be known there was a problem and the patches came out, and he was going to release at Black Hat. And two weeks later it had been reverse-engineered, figured out, and it was deployed. Well, this is far more critical than a DNS spoof. This is a serious problem with TCP and what these guys have found, which - and I haven't even verified it. But I understand from what they've said what the problem is. And I'm going to explain it during this next hour.

Leo: So before we get to sockstress - one word, sockstress - let's talk a little bit about last week's show and any security news you have. Anything going on out there?

Steve: Oh, yeah, bunch of stuff. There was an article that the BBC published, a news blurb that caught my attention. I thought - I was just kind of thinking, oh, goodness. Turns out that somebody bought a Cisco VPN router off of eBay that was still configured with the way it had been set up when it was sold. So he takes it home, plugs in it, hooks it up to his network, and he is connected into the internal private network of the VPN router's prior owner.

Leo: Oh, whoopsie. Oh, dear.

Steve: So the story reads, "Andrew Mason, from security firm Random Storm, bought some network hardware" - which happened to have been a Cisco VPN router - "from auction site eBay," okay, for, like, less than a pound. He paid nothing. He paid, like, scrap money for it. "When he switched it on and plugged it in, the device automatically connected to the internal network of Kirklees Council in West Yorkshire. Kirklees Council called the discovery 'concerning'" - it's a bit of a concern, yes - "but said that its data had not been compromised." So anyway, just a little heads up. You definitely want to remember to wipe the memory and configuration of anything like that.

Leo: Is it pretty easy to do? Must be, I mean, there must be...

Steve: Oh, yeah, you just return to factory settings, and it would forget everything it knew. I mean, obviously it had the IP address and the VPN keys, the security keys for their VPN server, and just linked itself into their network and said, ah, now you're part of our network.

Leo: Well, you know, I do that - I don't know if this is - I guess it's related, with Secure Shell, with SSH. It's secure if you set it up so that you just log in, you don't pass a password along, it just recognizes your thumbprint; right?

Steve: Yup.

Leo: But if you give somebody that machine, and you don't erase that stuff, it's just going to log right in, say hello.

Steve: Yes, yes, yes, yes, exactly.

Leo: I'm here.

Steve: In two other little blurbs, I wanted to mention that a lot of our listeners wrote in to inform me that Phorm was back, the P-h-o-r-m.

Leo: Oh, no.

Steve: It turns out that BT, British Telecom, is going to fire up another Phorm test, this time with opt-in, which is all anyone wants. There's still some concern that most users just click "Okay" without reading the text. But now Phorm has said that they're going to do a test. They want to find - they're going to select basically at random 10,000 users. And so those 10,000 people, when they attempt to get on the 'Net, their browser will be intercepted with an intercept page. This is exactly what I was describing when we were discussing this months ago. I said, look, this is all they have to do to make it okay.

So their browser, which attempts to get to the 'Net, will be intercepted because of course this is all network interception technology that Phorm is bringing. And so there'll be a page that comes up and offering them to opt in. Apparently now Phorm is attempting to promote this service as a benefit to the customers by saying, oh, that we're also going to give you antiphishing protection, that is, if you try to go to - since we're monitoring everything you do, if you go to a site that is a dangerous site, we'll give you some sort of interception warning. So they're promoting it as we're going to enhance your web security and blah blah blah. But at least it's an opt-in now instead of it being surreptitiously behind everyone's back.

And you know that our U.S. Congress has raised the alarm and said, wait a minute, folks, we need to look at this whole notion of third parties being involved in monitoring ISPs' traffic. So Verizon, AT&T, and Time Warner Cable have all stated to Congress that, if they ever do this, they promise to make it an opt-in system and never do it without their subscribers' knowledge or permission. Again, the problem is that so many people are just used to saying, okay, fine, clicking on anything that comes up that stops them from getting to the page they want to, that they may not read the fine print. But this is certainly all you could ask for from an ISP. I mean, you might want them to have another screen come up and say are you sure. But that seems unlikely.

Leo: So you're content with this. You think that this is adequate to make Phorm acceptable.

Steve: Well, you know, the problem with Phorm, remember, is that it is really nasty in terms of what it does with cookies. It inserts their own cookies, in addition to any website cookies, to any place you visit. So, I mean, it really messes up the computer of any client that filters through a Phorm-using ISP. But again, people will - I hope they explain that they're using Phorm, that they're, like, saying we're going to be using the Phorm system. Because the point is it has gotten such bad press in the U.K. that a lot of people would know, well, maybe I don't want to do that. I don't want to click on Yes.

Oh, and you can opt out. When you opt out it puts opt-out cookies on your system. And the question would be then is it going to put an opt-out cookie for every ISP? If the technology functions as we described it, they would need an opt-out cookie for every ISP. So even if you opt out, you would still be gunking up your system. You'd just be gunking it up with please-don't-track-me, please-don't-track-me, please-don't-track-me cookies, instead of this-is-who-I-am cookies. So the whole idea just is really distasteful. And it's unfortunate that this is the way they've implemented the technology.

I did see one blurb that said that they've determined, however, that the load on their system from all this filtering would require, in the case of British Telecom, as many as 300 additional servers in order to handle the additional traffic. I mean, all of this bouncing around and redirecting and cookie insertion and deletion and all that nonsense, I mean, it does pose an overhead. And that does imply also that customers are going to see some reduced performance. Because if any ISP needs an additional 300 servers, that says that those servers are busy involving themselves with traffic that they're not having to now, which says, okay, there's going to be some overhead for every single TCP connection you establish with a remote web server. The only good news is that SSL connections are still safe. They're not intercepting any sort of secure connections. So it sort of says get a secure connection every time you can because then you're not being filtered in any way when there's no opt-in or opt-out nonsense.

Leo: Is that because they can't?

Steve: They really can't, yes, they can't. I mean, the day that our ISPs require us to accept their own certificate authority so that they can terminate the SSL connections and filter that, okay, that's the point where you just say no, that's really not okay. There are, of course, corporations who are doing this, but that's within their own domain. I think that's...

Leo: Well, and ISPs could do it by redirection; right?

Steve: Well, no. ISPs, in order for it to function, they have to terminate the remote connection to the web server and then set up a new SSL connection.

Leo: A new certificate; right.

Steve: So they would have to give us a browser cert that said that their servers are authorized. It would be a mess. But it can be done.

Leo: So in fact that's what Opera Mini does, as I remember. Opera Mini, we learned, is not secure with SSL connections because, as part of their proxying scheme...

Steve: Exactly.

Leo: ...they do exactly that. They have their own certificate.

Steve: And my last little bit of news, I thought it was interesting that RealDVD has already been hauled into court by Hollywood. You'll remember that RealDVD was a new piece of software being offered by Real which would allow people to decrypt and copy DVDs to their computers.

Leo: But, but, but keep the copy protection on it.

Steve: Uh-huh. And Hollywood has said, yes, well, sorry, but you have violated the DMCA, which...

Leo: Well, technically I guess that's true, yeah.

Steve: Yes, they had to reengineer it, and they had to decrypt it in order to then put their own encryption on it. So their argument was, but this is fair use. This falls within fair use. And Hollywood is saying, uh, we don't like it. So they've sued each other now. And in fact Real sort of did a preemptive suit, apparently filed a suit looking for a judgment on the fact that this was okay. And whereupon the Hollywood people said it's not okay, and they sued them for copyright infringement and violation of the DMCA.

Leo: Now, they may win that case because there's one other, I think, related case, and that was the DVD Jukebox case; right?

Steve: And that's what allowed them to think they might be able to get away with it. But it'll be interesting to see, once this is tested, how it comes out.

Leo: So this was a device that was a DVD Jukebox but didn't use the disks. It copied it to a hard drive. And the judge - they sued, of course. And the judge said, no, no, this is okay because it's not accessible in any other way, it's just it's on the drive. It doesn't give somebody access to the data. They still have to have the disk originally to make it, and they can't access the data in other ways. I guess the concern would be you'd go out and you'd rent a bunch of movies, copy them over, and then could burn the movie.

Steve: Ah, right, right.

Leo: That would be a concern.

Steve: We've had a bunch of security updates. Firefox has moved from 3.0.2 to 3.0.3 and fixed a number of things. Mozilla also had a round of updates. And Java for Mac had a big, 136MB update. In this case it was fixing two serious vulnerabilities which had remote exploit aspects to them. So I just wanted to let anyone know who's using Firefox or Mozilla or Java that they're going to want to make sure they're running up-to-date code because those have all had significant fixes.

Leo: It's amazing.

Steve: I did want to mention also, just acknowledge to our listeners that you and I and Dane and Tony and everybody know about the half cut-off of last week's Episode 163. It caused some confusion, and I heard from a lot of people that they only got half the podcast. And...

Leo: Yeah. Thank you, actually, for bringing that up. So I guess we inadvertently cut off the upload, and so only half the podcast was there, actually.

Steve: Right.

Leo: And what we do in a case like that, just so people understand, we can fix it, but we are on caching servers. So AOL uses Akamai as a caching server. And Steve, actually I'd love to get some insight from you or maybe one of our listeners on this. You know, I would think in theory a cache - the way a caching server works is there's a primary server, which is what we upload to. And then when you request it, your request is actually forwarded to a server that's geographically near to you. That server, if it has a copy of the file, will just serve it directly to you. If it doesn't, it will go to the primary server - it's kind of like DNS - get it, download it, and then serve it to you. There are some problems with this, though, because while you would think a caching server would be smart enough to do some sort of hash or MD5 hash or CRC or something to say...

Steve: Even a timestamp.

Leo: Yeah, is this the same file. Apparently it doesn't. If the filename is the same - at least this has been our experience. We can't quite figure it out. If the filename is the same, it just says, oh, I've got it. And so if we do - if there's a mistake, which happens frequently, or we cut something off, sometimes the server that's the caching server itself will get a bad copy for whatever reason. And it will continue to serve that bad copy. So from time to time I'll get reports. And it happens, for instance, to the Akamai server in Florida more than others. So I'll say, when somebody says I got a truncated version, and I'm only hearing it from a few people, I'll say where are you, and they'll say often the same location. That's because your local caching server happens to have a bad copy. The only way we can figure out to fix this is to post it a second time with a new name. So what we do is we append a letter, usually "A." Unless we make this mistake more than once, just "A." So it was SN-16, whatever it was, 2A...

Steve: Yeah, SN-163 was last week.

Leo: ...3A.mp3 instead of just 163.mp3. And in order to do - the reason we do that is then we change the feeds; we change the link on the website. The caching server now gets another copy with a different name. It thinks it's a different file, which it is, and will serve you that one. So if you accidentally - and we fixed it within 12 hours. I got enough emails the first thing in the morning, I went, whoa. It happened overnight, unfortunately. So what we do then is we put up the second version. But you may have already downloaded the first version. So that's where the confusion probably comes from with people who got two copies of 163. Listen to 163A, that's the full version. 163 is damaged. I do apologize for wasting your bandwidth. Just it was a mistake on our side.

Steve: Also I have a little bit of sci-fi news. I know that we have a strong sci-fi subculture among our listeners. So I wanted to mention that Fox's new "Fringe" series continues to be really fun. I notified our listeners about it beforehand, I think. And I just continue to enjoy it. And I also wanted to give everyone a heads-up that the reason Amanda Tapping has left, completely left the Stargate series is that she's moved to a new show that is premiering this coming Friday, which will be the day after this podcast goes live, on Sci-Fi Channel, called "Sanctuary." And so the two-hour premiere of "Sanctuary" is airing. The show is interesting from a technical production standpoint in that it is massively virtual. I've seen a little bit of "Making of Sanctuary" stuff. They've been teasing us during some other sci-fi shows. And it's, for example, showed some of the actors standing around in just this room of green screens where everything was matted onto these green screens. And so all - they're doing all their acting and motion stuff with no scenery at all, which is all added later. And I know nothing about the show except that I think the sanctuary is a sanctuary for monsters.

Leo: Uh-oh.

Steve: Of varying descriptions. I mean, like, you know, so this is like where monsters go to seek sanctuary.

Leo: Cool.

Steve: And the plot goes from there. So it does sound like it could be fun. And there's another show on the BBC which I've been enjoying, "Primeval," which is also taking advantage of a lot of CG stuff, computer-generated graphics, like for all of the monsters. And I have to say I'm really impressed with how the technology has evolved so that weekly shows are now able to offer us the kind of things that we used to need to go to feature films to see. I mean, it's not the quality of high-end feature films in terms of special effects. But, I mean, it's very passable for a weekly. And it certainly gives us sci-fi fanatics something more.

And, finally, my last little tidbit is that I just happened to go over to the website of one of my favorite authors, Michael McCollum, who I've talked about for years. He's the guy at SciFi-AZ.com who does the eBooks and also publishes paperbacks. He has been really gratified with the attention that I've brought to his work to our listening audience. And I was just sort of curious. I go every few months when I wonder what's happening with the third book in the Gibraltar series. And I was met with some delightful news, reading from his page. He says, "After a too long gestation period, 'Gibraltar Stars' is well launched and moving steadily forward." His first one was "Gibraltar Earth," and then "Gibraltar Sun." I've read the first one twice and the second one once because I always like to start at the beginning and reread the series when a new one comes out so I'm all back up to speed. And this is just a fantastic trilogy, or soon to be a trilogy. Right now it's the first two books of the trilogy. And he said part one of the series is complete. It's launched and moving steadily forward. And he says "I am working on Part 2." And I can't read it, well, okay, this page says, "The Broa have discovered that there are wild bipeds loose in their empire and they are not happy. This will cause our heroes big problems in the future."

The series is really fun. Essentially we discover that we, by some coincidence, haven't been noticed by a vast - we, humanity, humans - by a vast sovereignty, I mean, a vast empire of really not very nice aliens who enslave all the races that they encounter. And by some coincidence our expanding radiosphere hasn't yet touched any of their listening posts. And anyway, oh, it's just wonderful, you know, space opera. So I'm delighted that he's working on the third book.

However, he says, "As I have long noted in the FAQs at SciFi-AZ, this website is an experiment. I've always maintained that people are basically honest, and when they understand the damage done by giving out my books for free, they will not do it. The purpose of this progress report is to let those of you who have read 'Gibraltar Earth' and 'Gibraltar Sun' know that the end of the series is in sight. However, if this trading of my" - oh, I'm sorry, I skipped a paragraph. He says, "And now for the bad news. Today a reader alerted me that there are unauthorized copies of several of my books on the web." One of the nice things he's done is he does not copy protect these. They're in various eBook formats. But it means that they could be uploaded by people who have purchased them from him and shared, which of course is a bad thing to do. He says, "I checked, and the postings appear to be the work of two individuals, both of whom probably feel that they are just being friendly users of the Internet and sharing work they enjoyed with others."

Leo: So he puts - he encodes the PDF with information about who bought it.

Steve: Either that, or he's just looking - I don't know that. Maybe he does.

Leo: I would.

Steve: Yeah, I would. I mean, that's what I do with SpinRite is I just put the name of the user in the product. And I say, look, you know, don't share this because that's - I cease to exist if it is shared. So anyway, so he says, "However, if this trading of my work becomes epidemic, it totally destroys my incentive to put another 5 to 600 hours into the new book or write follow-on books. If I can't sell my books, it makes more sense for me to become a full-time publisher at Third Millennium Publishing. If you become aware of people posting my work and have access to them, please explain the damage they are doing and politely request that they take the books down. The problem with 'trading' creative works is that you can save a few dollars today; but, ultimately, you will kill the goose that lays the golden egg." And his stuff is not expensive. And it's really good. So anyway, I wanted to give another plug to this Gibraltar series. I am so excited that I'm going to find out what happens next because he has just written a really interesting yarn here that is going to take us in some fun direction.

Leo: Yeah, you know, it's funny, I read the first one, and I've just gotten behind because of Peter F. Hamilton.

Steve: Oh, yeah, that'll slow you down.

Leo: I'm slogging my way through the Night's Dawn trilogy. I'm about halfway through now. But it's just long.

Steve: Oh, you're reading Night's Dawn.

Leo: Yeah.

Steve: Yeah, well, it does get, believe me, it does...

Leo: That's a very long story.

Steve: Yeah, it is. And it's fun. I really liked it. But I have to say I ended up - I don't mean to spoil it for you, Leo. But it really seemed to be overly long. And I thought the first book was the best of the three.

Leo: Yeah. I got, you know, I bought it on the Kindle. So I can't - it's all together, smooshed together. So it's just like one...

Steve: Ooh, big. In fact, remember that the Sony Reader couldn't even load it. It crashed the Sony Reader. I did an experiment when the Kindle first came out and specifically bought it for the Kindle just to see if it would work.

Leo: It's huge.

Steve: Yeah. But you know how the Kindle shows percentage dots. You probably haven't seen many of those.

Leo: No. I know I'm halfway through because the dots are now almost halfway across. But, boy, I mean, so that's - it is, it's very, very long. But, I mean, it's fascinating. But it's kind of slowed me down on science fiction. I only, you know, I read other stuff as well as science fiction. And so I'm not making as much headway as I should. Hey, we're going to...

Steve: Well, I have a - go ahead.

Leo: You've got a SpinRite story, I'm sure. And I know we want to talk about stress to my socks.

Steve: Well, that's the big topic, yes.

Leo: So, Steve Gibson, have we got all the errata out of the way? Anything else we want to talk about?

Steve: Well, we have SpinRite. I actually ran across this report just this morning. And, in fact, I settled down, and I was reading through my mail bag, all set to do a Q&A.

Leo: Right.

Steve: When I got a note sent to me by one of the people in the GRC newsgroup that was just forwarding me a blurb that had been posted in one of our newsgroups. We have a group that I named "linkfarm" because it's just - it's where people post links to other things. And that's where this notice of this sockstress problem came up. And so as I pursued that I realized - and listened to the audio file - it's like, oh, no. We have to talk about that today. So that's when the Q&A got postponed till next week.

But prior to that I had run across this posting that was just sent to me - it's dated the 30th, so yesterday - from someone who called himself Matthew, with a subject "SpinRite Saves a Tech from Himself." And he says, "Hello, Steve and Leo. I just have to start by saying how much I enjoy your netcast. I started listening when it was a mere 32 episodes old and soon went and started from #1, as you often suggest listeners do. I listen to most of your other netcasts, Leo, and have even branched out to others. But I always come back to Security Now! as my favorite. There is something about the quality of this netcast that is just precise. I have an hour drive to work, which I'm afraid has nothing to do with traffic. And Security Now!, well, Security Now! and Audible has made a very boring drive very enjoyable. I actually like getting into the car.

"Steve, you've mentioned often that you do not tire of SpinRite stories. And while mine is not so much exciting as a Navy SEAL rescue mission or as honorable as saving a family's lifetime of photos, it is my little SpinRite story. I'm a big fan of SpinRite and have owned a copy for, well, I guess just about two years now. I work for a local college in Southern BC and spend a lot of time dealing with staff, faculty, lab and student machines with various issues and problems. My first move is always SpinRite. When a machine is brought in, SpinRite. When a client is complaining about their system, I tell them to buy SpinRite. When it's time for lunch, SpinRite. Okay, maybe not that much SpinRite, but close. And that's true of just about any machine I work on. While I admit a lot of the time the problem is not a hard drive, it makes me feel better running your tool. It also gives me a chance to do a little research on whatever other symptoms the machine might be having.

"So over the weekend I moved into a new place, after which my own PC would not boot into Windows. It did that great little trick of an endless boot-crash-boot loop that I know you're familiar with."

Leo: Ugh. I hate that. Yeah.

Steve: "In panic, I grabbed my kit of software and tools and went to work. I tried every trick I could think of - running the Recovery Console, the Repair Windows, reinstalling over top of my current installation. I tested memory, the sound card, the video card, well, you get the idea. And yet still the machine continued to boot loop. Now, I don't have anything really all that impressive on my machine. I actually reload my PC a lot, so it's not a big deal. I'm not a torrent junkie. I don't have lots of data that I can't afford to lose. I do enjoy a challenge of fixing a problem. But after four hours of tech time on this machine I decided I would just blow it away and start over fresh.

"So I grabbed my CD case and started flipping through it for my copy of Windows XP. And suddenly I stopped, staring directly at the SpinRite boot CD. I looked at it, a bit surprised. I hadn't even thought of running it. I laughed at myself and threw it in the machine, started it up, and walked away. When I came back I found SpinRite had happily finished its task. So I rebooted the system, and what do you know, it booted straight in. And there you have it. Another testimonial of the great power of SpinRite. Keep up the...."

Leo: Wow, that's really interesting. Wow.

Steve: So I got a kick out of the fact that here he is, anytime anyone else tells him, oh, my machine is doing something strange, okay, run SpinRite. I'm hungry, okay, run SpinRite. But when it happened to him, you know, he grabbed all of his own tools and thought, oh, okay, I'm going to fix this, instead of just running SpinRite and letting it fix the problem for him. So...

Leo: There you go.

Steve: Got a kick out of that.

Leo: It's the same with me, you know, it's the cobbler's kids, you know, you always forget for yourself. You can do it for your client. You know what's the right thing to do. You just forget. Boot loops. We have to do a thing on boot loops one of these days.

Steve: Okay. Well...

Leo: All right. Subject at hand is better than boot loops. It's sockstress.

Steve: Okay, so here's the story. Two Swedish researchers, Robert E. Lee and Jack Louis - actually I had his middle initial, but I took it out because, you know, of course Robert E. Lee is...

Leo: That's the general, yeah.

Steve: Exactly, the famous general in American history. They have a security research site called Outpost24. And they're some security researcher guys. They explain in this audio MP3 - which is now being widely listened to all over the web. Slashdot picked it up. It's linked to in the anchor posting on Slashdot. It starts out in Swedish or Finnish or something-ish. Definitely not English. But about four minutes in it switches to English because I guess these guys speak English. And so the interviewer, who is saying something for the first four minutes that I can't understand, switches over to English. And then the rest is understandable.

I'm going to put a link to the URL in our show notes. And I will probably also trim the beginning of it and host the MP3 myself just so that people can listen to it in English from the beginning, and in case the site where it's coming from becomes overloaded. I don't know what kind of servers or bandwidth those guys have. But I have a feeling this is going to be a highly listened-to MP3. And unfortunately there's no doubt that it has already come to the attention of the bad guys, the black hats.

So what they explain is that somebody asked them to do some penetration testing of a large network. And so they needed to write a large sort of distributed scanner so that they could scan a large network at once. Well, scanning means that you create TCP connections, if you're doing a TCP scan, like for example ShieldsUP! does at GRC. Of course, I'm no newbie to scanning. I know all about how to do massive parallel scans. That's what GRC does.

So these guys did - they did what they call a "userland stack," which essentially means that - "userland" is the term, for example, as opposed to the kernel, where "userland" means that you're running an application. So they wrote a TCP stack as an application because they wanted to write a special one rather than relying on the kernel and just opening a connection through the kernel. The reason they had to do this is that they wanted to be sending out, essentially, so much probing traffic that they would crash their own kernel if they were to do that. So instead they said, okay, we're going to write our own TCP stack.

Well, now, this is really not difficult to do. There's tons of source code on the 'Net. I mean, basically any Linux source is a TCP stack. And, for example, if you want to check to see whether a remote machine will accept a TCP connection, you just send a SYN packet. And that's trivial to do. And, for example, Nmap is an example of a very popular, widely known, application-level scanner which does this. It implements a userland stack, essentially.

Leo: Oh, I didn't know that. I mean, I've used Nmap many times.

Steve: Sure. And so basically all you need is raw sockets. Here we have again raw sockets. This is not something which Windows would allow you to do. But any Linux or UNIX machine or any Windows server would allow you to just generate raw packets yourself. So you send a SYN packet out, and the far machine sends you back a SYN/ACK, which is acknowledging the receipt of your SYN and sending you its SYN, which you then acknowledge. And that's the famous three-way handshake.

So these guys said, okay, we want to be testing a whole bunch of machines at once. So we don't want to need to maintain state. That is, we want to have so many outstanding scans in progress that, when we get the SYN/ACK back, we're able to, from that, verify that this is a SYN/ACK from a SYN that we originally sent out. What they call this is "client-side SYN cookies," or "client SYN cookies." And we've talked about what SYN cookies were once before. SYN cookies were an innovation, I think in, like, 1999. I believe it was Dan Bernstein. Now, this is the same Dan Bernstein who we've talked about recently who realized that there was a problem with DNS spoofing. Dan is a smart security guy. And there was a problem...

Leo: Dan Bernstein or Kaminsky?

Steve: No, Dan Bernstein. Yeah, Kaminsky, of course, is the hacker who realized what the problem was in DNS. Dan Bernstein is the security researcher who said source port randomization for DNS servers...

Leo: Oh, he predicted the problem, yeah.

Steve: Yes, yes. And he has his own DNS server that he put together with the highest level of security focus which was never vulnerable to the DNS spoofing problem because, he said, 16 bits of entropy, which is contained in the DNS transaction ID, is not sufficient. So use that and a random source port and you get 32, which is, you know, makes spoofing enough more difficult that it's probably no longer practical.

So anyway, Dan said, okay, back before this there was a problem with what's called "resource consumption." The original denial of service attacks exhausted the resources on the machines. They were not bandwidth-flooding attacks, which is what we have today. They were resource consumption attacks. Here's what was happening. An attacker would send a SYN packet to a server. The server would say, oh, somebody wants to connect with me. So the server would allocate some buffers to receive the data, and some other management space, basically commit some RAM to dealing with the connection that was in the process of being established. Then it would send back - it would generate a random number to encode or to contain its sequence number, which it would store in the state tables. And then it would send back the SYN/ACK.

Well, now, normally what would happen is that the person initiating the connection would respond with an ACK. But in the case of this being an early form denial of service attack, instead the attacker would just keep sending SYNs. The SYN/ACKs would come back, or maybe not even back because they could spoof the source IP. So this person would be sending SYN/ACKs out to random places on the 'Net. But the point is every single time a SYN packet came into the server, it believed a valid connection was going to be established, so it would allocate a chunk of RAM, and another chunk, and another chunk, and another chunk, and so on.

And it turns out that the early PCP implementations had no protection from this. And you could simply cause them to consume either all of the machine's RAM or all of the RAM that the stack had been allocated. And suddenly it couldn't accept anymore connections. It was busy waiting for the ones that were so-called half open to complete their opening process, which never occurred. And as soon as you did that to a commercial server that was online, it was in denial of service. It simply couldn't accept anymore connections. Sometimes it would lock up the whole system. Sometimes the stack would freeze or go sideways. I mean, all kinds of bad things happened when this occurred.

So the notion of a stateless connection acceptance evolved, and it was Bernstein who actually invented this first. By bizarre coincidence, I came up with the same idea later. I don't remember when it was. I think it was I was working on a client for ShieldsUP!, and I was concerned that, by accepting these connections, I could be subjecting myself to a denial of service attack. And so I independently came up with the idea of a stateless connection acceptance which I called "Genesis." It was an acronym for something that the folks in the newsgroups came up with, which was neat.

Anyway, the way that works is, the idea is you want a server to be able to accept a SYN packet and send back a SYN/ACK packet, so that it will then wait for the following ACK to accept to complete the three-way handshake. But you don't want to require it to have to remember anything about this pending connection until the final, that third ACK packet comes back to it. So what Bernstein cleverly realized is that there's enough information in the SYN packet with the source IP, the source port, the destination IP, the destination port, and the initial sequence number, that there's enough bits of entropy that the server could send an ACK packet back which cryptographically encoded that information in the incoming SYN such that when its SYN/ACK packet came back to the person wanting to establish the connection, if it was a valid person, when they acknowledged the receipt of the server's SYN/ACK, the server could independently verify the information contained in that final ACK of the three-way handshake and get total confirmation of the fact that a three-way handshake had occurred because only it could have generated the SYN that matched the remote port and IP of the person wanting to connect to it.

So essentially this immediately solved the problem of this resource consumption denial of service attack. So that's what SYN cookies are. The SYN cookie is the name for the idea of encoding in the SYN packet from the server a cookie, which is actually a cryptographic combination that represents the data that was sent in the original SYN from the client that wanted to connect to the server. So denial of service attacks, as we know, have not gone away. They've changed.

Leo: So is everybody using SYN cookies?

Steve: Yes.

Leo: Oh, interesting.

Steve: Linux has it. Windows has a version where, if it sees this happening, it can kind of switch on SYN cookies. There are some mild compatibility problems with SYN cookies because there are features in TCP which are conveyed in the initial SYN packet, things like option bytes and window sizes and some things that you'd like to have the server remember. But if it's going to be in a SYN cookie mode, where it cannot afford to remember anything from the incoming SYNs, then there were some variations later that tried to encode some of the more critical aspects of that in the SYN cookie so that that could survive. But in general there are some mild compatibility problems. So you'd rather not have them on all the time. And in fact Windows adaptively turns them on when it notices that the server might be under a SYN flood condition, if it sees that its stack's resources are in trouble, like lots of half-open connections, as they're called. It'll hit a threshold, and then it'll switch into a SYN cookie mode where it'll give up those features which it would like have, but which it can't have because it just can't hold onto state after receiving a SYN.

Okay. So what these guys did is they sort of did the same thing. Now that everyone understands what a server-side SYN cookie is, these guys did something similar on the client side. They would emit SYN packets into this vast network that they were scanning, and they would remember nothing in their own app because when the SYN/ACK came back, that would tell them that it was a SYN/ACK from a SYN that they had sent, and that they would then send the final ACK in order to complete the connection.

Now, what they describe is, and as I'm listening to this audio file from them, I mean, I've explained this in far more detail than they did. But everything I've explained is completely understood by everyone who really understands TCP and has been around for a while. So none of this is news. What they then - so as I'm listening to this, I'm thinking, okay, stop talking. Stop the music. Stop the MP3. Don't say anything more. But they kept on going. And they explained that in their testing they inadvertently started crashing machines, that when they were doing this, machines started - and, like, routers.

Leo: This is interesting. So it's just an accident that it was discovered.

Steve: This was an accident.

Leo: Wow.

Steve: And these, you know, some machines would become nonresponsive. And they talked about dropping packets and that, like, in cases where packets were dropping - and, I mean, I instantly knew what it was that was going on. And it was like, okay, stop talking. Stop talking. Don't say anything more.

Leo: Is this because this was a theoretical possibility that you were aware of? Or just a light went on for you, or...

Steve: No, it's because I really understand how these protocols work. And I've written some TCP stacks several times myself. ShieldsUP! is a full TCP implementation; and I've written several, several times. And I've written DNS servers. I just finished writing one for this DNS spoofing test that we'll be talking about soon. And so then they talked about timers. And it's like, oh, no, no, no, don't you...

Leo: Stop. Stop.

Steve: Don't talk about timers, please. And they just gave it away to anybody who understands TCP. I mean, I'm, you know, unfortunately the world is now full - well, not unfortunately. I mean, it's a good thing. The world is full of people who understand TCP. But the problem is, not everyone's motivation for having an understanding of TCP is the same. I have never had any interest in bringing down foreign machines. I could do so easily, but that's not been my goal. There are people who, now that this has come out, will listen to this audio. They will know, exactly as I did, instantly, how to do this.

Leo: So this wasn't an attack that you had thought of before. It was a different attack, something that you hadn't thought of.

Steve: Well, I just - my mind doesn't think about attacks.

Leo: But in other words, if you had been thinking along those terms, this would have - how obvious would this have been?

Steve: This is so obvious.

Leo: It's pretty obvious, okay.

Steve: Yes. So for example, I'll give our listeners an example of one. And, okay, so they talk about resource consumption at the server. They wondered what was going on. They pursued it. And they have figured it out. So they have something called "Unicorn," which was their original scanner. And then they have something that they developed called "Sockstress." Sock, of course, is as in sockets, which is the universal term for TCP sockets, which is sort of - a socket is an abstraction of the IP address and the port number and the state of an IP endpoint on the Internet. So they were talking about timers and about reading the Linux source code and seeing comments in the source code where the original author of the Linux stack, I don't know if that was Linus or who it was, but said okay, now, this is something we want to make sure we don't get too many of. And so they're thinking, oh, well, it's easy to give the stack too many of those. And so what they're claiming is, and I have every reason to believe them - as I said, in a couple weeks they're going to be demonstrating this. They're claiming that they can bring down any server that they have aimed this tool at so far - Windows, Linux, BSD, and apparently routers. So anything with...

Leo: That's interesting. Routers, too. That's not good.

Steve: Well, because routers have open TCP ports. For example, routers will typically accept BGP, Border Gateway Protocol connections, from anywhere because that's the way they exchange their routing tables. And so they say in this audio that in some cases it'll only kill one service. In some cases it will kill the entire machine. And they said in one case, and this has been repeated in text that I saw in several postings, that the machine would no longer boot after they did this to it.

Leo: What?

Steve: So, okay. So here's an example. Remember a while ago there was this notion of tarpitting. When we had, like, Code Red and Nimda were scanning for vulnerable Windows services, there was this notion of tarpitting. The idea would be that you would be just a random end-user who would want to hurt the people who were scanning you. So you would deliberately open ports, and you would run this tarpitting software. What the tarpitting software does is when an incoming SYN packet comes in, it accepts the connection, just like any TCP stack would, sending back a SYN/ACK. Then the attackers says, oh, I got somebody here, and it will ACK back to finish the three-way handshake.

Now, what you've established at that point is an asynchronous connection between the two endpoints. That is, either end can send data to the other. In TCP there's something called the "window." And the window is essentially "advertised," is the jargon used. It's advertised in every acknowledgement packet that goes back to the other end. What you're doing is there is a field called a "window" which says this is how much buffer space I have at my end. And so as acknowledgements come in, each of those acknowledgements is telling the receiver of the acknowledgement how much buffer is currently available at the other end. And that allows that end to send data, saying okay, the other guy has told me he's got 16K. So I know that I can send as much as 16K without any additional acknowledgement. And so as data goes out, for example, and acknowledgements are coming back, that window size may decrease because there may be less buffer.

So what this is, it's very clever. It's part of the original TCP protocol. And it allows each end to send ahead, that is, to send data in advance of having it acknowledged because each acknowledgement says, well, I'll guarantee you that at least this much buffer is available at the time that I've sent this acknowledgement. Well, if you send an acknowledgement packet that says I have zero buffer, what happens is that is essentially saying something's happened at my end. Somebody, the client, has not taken the data out of the stack. So I'm waiting for them to do that. In the meantime, you can't send anything more because I've got no buffer. So it stalls the sender from sending anything.

Well, what the sender does is starts a timer and every so often does what's called a "window probe." It sends an acknowledgement to the other end in order to get it to acknowledge. Basically it acknowledges one less than the number of bytes that have been received, which stimulates the side that is claiming it has no space available, stimulates them to acknowledge essentially correcting the sender's knowledge of where they are in the communication, saying wait a minute, you're one byte behind, you really mean this. In the process, the acknowledgement contains the current window size. So essentially it's a way for one end to probe the other, asking do you have any buffer space yet. I've got all this stuff I want to send you. Do you have any buffer space yet? The other end keeps saying no.

So in the tarpit system what this would do is, as soon as the connection was established from the remote server that's trying to attack you, the sender would send back an acknowledgement saying I've got no window space. Well, that would hang the connection. So the idea would be, if a whole bunch of people out on the Internet, in the tarpit case, if a whole bunch of people ran these tarpits, then the scanners, the Nimda and Code Red scanners, would end up collapsing because they'd be spewing out SYN packets, trying to find vulnerable systems. These pseudovulnerable systems would send back acknowledgements saying that, yes, I have a service. Let's establish a connection. But, oh, by the way, I've got no buffer space, so hold on. So what would happen is these Nimda and Code Red scanners would end up collapsing because they would have too many connections in this state where they were valid connections, but they were never able to send any more data, and it just locks them up. So the important point here is that in order to get permission to send data, they have to set a timer which periodically probes the other end to see whether any buffer space is available.

So with that, with the understanding that's the way TCP works - and TCP has a whole bunch of other timers. For example, one is dropped packets. And we've talked about this before. If a packet is lost, then there's no knowledge because the Internet has permission to lose packets at any time. If a packet is lost and the sender of the packet doesn't receive confirmation, it'll time out and resend. And then it times out, like, twice as long and resends again, and then twice as long and resends again. So those timers are consuming resources.

So it's very possible to establish a connection with a server; and, in doing so, you've established a real connection, that is, you're a bad guy, but you don't have to worry about hiding because, for example, you're in some bot army. You're a zombie computer that's been taken over. So you establish a connection with a computer, and you then do things that cause the remote end to consume resources, like timers. And if you are using a so-called "userland stack," which really means just if you are generating packets yourself, your computer is not actually your computer's kernel, your computer's TCP/IP stack, which would otherwise be similarly vulnerable because, I mean, both ends would be consuming resources. You're not consuming any resources at your end because you're using this notion of client SYN cookies so that you're not having to maintain all this state. You encode in the communication the information that you need in order to, for example, continue sending back an ACK saying, oh, sorry, we still don't have any buffers. Meanwhile you're sending out more SYN packets. You're accepting the SYN/ACKs and sending back the ACKs saying, oh, sorry, that's a good connection, but we've got no more buffers. So this is, again, this is a return to the original style low-bandwidth denial of service attack. And assuming, I mean, so...

Leo: How many different connections would you have to open to bring down a typical server? Obviously one's not going to do it; right?

Steve: No. One won't do it. In one blog posting by Robert Lee, he said that their attack was rate limited to nine packets per second, distributed between 16 different source IPs.

Leo: That's nothing.

Steve: So effectively less than one per second per source IP. As you said, Leo, that's nothing. Here's the point, is this is like a dribble attack. You could just be sending out - you send out SYN packets; you accept the connections; and you say, sorry, we have no buffer space. And you just keep doing it. This passes through bandwidth limitations. It passes through the normal flooding-style denial of service, the SYN-spoofing denial of service attacks, because you're creating valid connections at the other end and then tying up resources at the server's end, consuming none at your end.

Leo: This is scary.

Steve: Yeah, I mean, I'm disappointed because - okay. First of all, I've just, again, I haven't tested any of this. I've just articulated an example of how this might be done, which will be absolutely obvious to anyone who listens to this audio who has the capability that I do of understanding TCP and generating their own packets. And as we've seen with the DNS spoofing and the nature of Internet attacks now, this will be done. There will be some, I mean, just as a challenge to demonstrate for proof of concept's sake that, oh, look, I can crash any system that I want to. These guys say that there are different vulnerabilities in different stacks; so not all stacks, that is, not all Windows and Linux and BSD and routers are the same; but that they have never encountered one using this notion of a post-handshake protocol abuse, which is what I would call it. The idea is you do the TCP handshake in order to get real state now being saved on the server side. Then you do things that cause it to burn up resources. And, for example, stalling the connection by sending back a zero window, you know, that's the first one that occurred to me because it is the tarpitting technology that we've seen before, and you can just stall the connection. And who knows which one, if any, of the servers are vulnerable to that? But you could also do things, essentially just not letting the connection progress and requiring the system to use timers. And they specifically mention timers as something that is a limited resource. And it's understandable that it would be. But this apparently destabilizes every machine that they have come across in different ways.

Now, given that that's the case, I can't understand why they couldn't demo this to Microsoft and Cisco and, for example, essentially the equivalent of the people that Dan Kaminsky demonstrated his stuff to and make it clear to them why this is such a problem. They also talk in their audio file about where there are firewalls and, for example, load-balancing systems in front of many servers, which are now common, where there's like a front-end appliance which is doing DoS protection and connection proxying or something, and then doing load-balancing distributing that, if they bring that down, and they have been able to...

Leo: Sorry about that.

Steve: No problem. If they bring that down, and they have been able to, then of course the whole server farm behind the appliance goes down. So, I mean, again, as I was listening to this I was just closing my eyes, thinking, no, no, no, don't keep talking. They're still sort of being coy about it, as if what they've just published in this audio doesn't completely describe to anyone who has an understanding of this how to perform the attack. It does. I mean, we will see this in a day or two. It's just, I have to say...

Leo: So and is there no, I mean, here I am, running a bunch of websites. Is there nothing I can do about it?

Steve: That's how I feel, too, Leo. No, there is...

Leo: It doesn't - and we should say that this is only impacting somebody who runs a website.

Steve: Right, very good point. Now, the reason end-users were so concerned about the DNS spoofing attack was that what that meant was that they could be redirected to a bad site by if their ISP's DNS server had its cache poisoned. In this case, end-users who are not running servers or any kind of services, who have no exposed ports, are safe. So, for example, if you use GRC's own ShieldsUP! service to check for open ports - of course that's what I wrote it for all those years ago - and you come back stealth, or you have no exposed public ports, you're okay. But what this potentially means is that anybody who is accepting Internet TCP connections, which is, I mean, even somebody who is, like, deliberately exposing some connections, an FTP server or a web server or something - not to mention any commercial sites that by definition are exposing these services or offering these services to the 'Net. Once this stuff gets out there, it means that any - and given that these guys are not exaggerating. And I believe what they're saying. They're going to be demonstrating it, as I said, in Helsinki in a couple weeks. They can crash the listening service or the stack or the machine, depending upon the nature of the resources that they're able to consume over on the server side. And they can do so with very little resources at their end. And that's the key. You no longer need a bot fleet. They say in the podcast that a - the example they gave that I read in a posting was this rate limited to nine packets per second. But they also said in the audio that a single broadband user has plenty of upstream bandwidth to bring down a major server. I mean, as strong a server as there is. And...

Leo: Because this isn't based on flooding the server, which requires a lot more bandwidth than the server has. This is based on screwing with the TCP stack.

Steve: Yeah. The way you can think of it is the very first denial of service attack was a resource consumption attack using SYN packets.

Leo: SYN flood, right.

Steve: And just the SYN packets would end up burning up connection resources. So we developed armor against that, the so-called "SYN cookies" and other kinds of stateless connection approaches, so that you were protected from a SYN flood. Okay, so but the problem was the stack behind, the rest of the stack behind that never got hardened. So what these guys are doing is they're saying, oh, let's go ahead and accept the SYN - we'll accept the TCP connection, establish it. Now, by the nature of TCP, the fact that you have to be able to retransmit lost packets, that means you have to have state. If someone says I've got no window available, you have to probe them, asking to see if they've now got window available. You have to have timers by the virtual definition of the TCP protocol in order for TCP to function in all kinds of different ways. So what they've done is they've come up with protocol, TCP protocol attacks on the protocol itself that so far they claim no machine is invulnerable to. And this is bad. I mean, and so this is why I said, okay, wait a minute. I think I got to, like, question #6 of today's Q&A, assembling the questions.

Leo: And then you said, whoa.

Steve: I said, okay, stop, hold on, wait a minute.

Leo: So, now, this wouldn't be so hard to fix. I mean, all you have to do is fix the TCP stack to do something with these window requests; right?

Steve: Yes. What will be necessary, and it's unfortunate, again, I just - I don't understand why they went public with this, but they did. What you would need is you would need to harden the stack so that it has some strategy for dealing with this. Now, the problem is, first of all, if this was one attacker, if it was one attacker attacking Yahoo.com, well, we've got his IP address. So it's easy just to immediately blacklist him. And so, okay, and not only that, but we know who he is. So you'd be dumb to do this because you're going to have the authorities knocking on your door before long. But now we're in a different era. We're in the era of bot fleets. And so this technology will then immediately move to the bot fleets, where...

Leo: So it's not the hacker's computer that's doing it, it's the computers they've co-opted ages ago.

Steve: Exactly. And so they've got 10,000 machines. And no longer do 10,000 machines have to do bandwidth floods. Now 10,000 machines can just all send out, you know, they do this stateless attack where they send out SYNs, they accept SYN/ACKs, and then they do whatever they do after the connection has been established. So they can all afford to establish lots of connections into a single server which believes that these are all valid clients. It's not clear that there's a technology that would differentiate from valid clients because the server is handling lots of valid clients, and here's just a bunch more that are not flooding them with bandwidth. And the things that these guys are apparently abusing are core requirements of TCP. Now, what you could do would be to look at hardening the stack, that is, you don't want it to collapse. You certainly don't want it to crash your machine. I think that case where Windows or whatever server it was that just would no longer boot, that just has to be an anomaly.

Leo: Yeah, because, I mean, how would that crash your machine and...

[Talking simultaneously]

Leo: ...going to have to destroy something. I mean, it's not.

Steve: Yeah, it would have to make an alteration, a bad alteration to...

Leo: Can't do that.

Steve: ...the hard drive. On the other hand, I could see where, depending, I mean, if this is the case today that TCP stacks are able to bring the machine down, you could imagine that it might have been writing to the swap file and written to the wrong...

Leo: Overwritten it or something, yeah.

Steve: ...space on the hard drive or something. So anything could happen in that case. So...

Leo: Let me ask some questions.

Steve: Yeah.

Leo: IPv6, does this change anything?

Steve: No. And in fact they talk about that they've - the interviewer during this audio gets as worried as we get, although he doesn't understand this nearly as well now as you do and as our listeners do and as I did when I was thinking to myself, okay, stop talking, stop talking, don't say anything more because you're going to - they did, they just gave it away. He didn't understand because he got confused with what SYN cookies was and so forth. And, you know, he's not a packetsmith. But any of the world's packetsmiths instantly know what this is.

Leo: And what to do.

Steve: And what to do.

Leo: To exploit it, yeah.

Steve: Which is the only reason I've been willing to talk about it is that this is...

Leo: They already know.

Steve: ...already out there. This is not giving anything away. And I don't even know, for example, that using a zero window size would cause a problem, but I won't be surprised if it turns out that's one of the things that you can do because it's just obvious. And again, that's the problem is this is obvious. And anyway, so...

Leo: So IPv6 doesn't - it uses the same stack, doesn't change anything.

Steve: Yeah. Well, and here's the problem, again, is that IPv6 fundamentally is still using TCP, and TCP has to have state and has to do things intelligently, for example, retransmitting packets when it hasn't heard back after a certain amount of time. And in order to know how much time it has to have timers. Now, it might be, for example, there are many ways to do timers. I've implemented timers in many of my systems which do not consume resources in the same way that, like, dumb timers might. So it might be that there are dumb things in the current stack implementations that could really be fixed and made much more bulletproof. But the time then to have done that was before going public with this, rather than now, because none of that work has been done. And this is going to take a chunk of time. And if these guys are right, we're talking about anybody with a listening TCP port is vulnerable to having their system taken down until the stack is hardened and fixed.

Leo: Randal Schwartz is in our chatroom, and he says that OpenBSD, they've already discussed this on the OpenBSD list, and in fact it's not an issue with the OpenBSD stack. So it is - and we talked about this before as being the hardened version of BSD. Apparently, if you are running a BSD, FreeBSD or NetBSD, you could copy the OpenBSD stack or, I guess, winsock, whatever they call it, the sock, and be okay.

Steve: Well, and we have to hope, then, I mean, that the OpenBSD guys who Randal is quoting know what they're talking about and understand everything about what these guys have done. Because these guys have not yet said exactly what they've done. They've not released Sockstress to the world. And so I guess - so the hope is that the OpenBSD guys have hardened their stack, as I've been talking about, in every way. I mean, so that OpenBSD can't be brought down. BSD itself, the Berkeley Standard Distribution of UNIX, was mentioned by these guys. I don't know which version of variant of BSD they were talking about. But they have said that they can bring down BSD.

Leo: Yeah, exactly. And of course the other BSDs are vulnerable. But apparently OpenBSD is not. They do something - they don't actually implement SYN cookies. They do something else which is kind of interesting.

Steve: Yeah, well, there are many - see, and here again we've got to be very careful because what I immediately saw, I mean, Slashdot, that is not known for its rocket scientists, they immediately went off sideways and had no understanding what they were talking about. This is not about SYN cookies at all. It's got nothing to do with SYN cookies. And that was one of the things that confused the interviewer in this audio file was that he got locked, latched onto SYN cookies and what were client-side SYN cookies and blah blah blah. This is, you know, it's not about what this is. So it's important for people to understand that this is about abusing the deeper protocol behind the original handshake. And that seems very clear from the audio that that's what these guys have done.

And so we just have to hope, for example, that the OpenBSD stack is as bulletproof as the OpenBSD guys hope it is. And it's a perfect example of what I meant when I said "hardening the stack" because you need to, like, look at it from a worst-case vision.

Instead of having comments in your code, as apparently there is in Linux, and this was the example given, where the author says, oh, we'd better hope that not too many of these happen, instead your code wants to say, and it doesn't matter how many of these happen. We've got it handled, baby.

Leo: We could have a problem if too many happen. Let's just hope not. Hope is not a good security strategy. So presumably Microsoft, Apple, and all of these guys are working on repaired stacks; right?

Steve: Well, that's one of the other reasons I thought this podcast was important was, since the cat is out of the bag, and the bad guys all know about, and Slashdot knows about it, and DSLR, DSL Reports knows about it. And SANS has a blurb about it. I mean, I'm hoping it's come to the attention of the people who we need to have fix it. If not, I know it has now.

Leo: And not hard to fix.

Steve: Well, we can't change the TCP protocol. But I would say probably not - I don't know what "hard" means. The problem is you don't want to destabilize your stack by making big changes to it. But you want to ask yourself, for example, in the case of all these timers or any other kinds of resources that can be consumed, what happens if we've got too many connections in this state? I think that would be the question. What happens if we have too many connections in this state? If it's possible to artificially put a stack's connection into a certain state, what happens if we then have a bunch of these? And I think that's where the vulnerability is.

Leo: Okay. And the patch would not be for end-users anyway, it would be for servers. It would be for...

Steve: Well, it'll be for everybody. Everybody's stack is able to receive connections. Normally people don't expose them. Thank god Microsoft finally put a firewall into XP and to Vista so that there aren't open ports exposed to the Internet. But what we will - every OS, and routers, I mean, routers are the other problem, too. Because, I mean, if BGP port is open, as it typically is for routers, and many other routers have other ports open, as well, if routers are vulnerable, if the Cisco stack in IOS or Juniper's stack is vulnerable, then this is a huge problem because routers are easy to find. A traceroute gives you the IPs of all the routers between here and there. And it's then easy to attempt to create BGP connections, which any routers that accept that, if they've got vulnerabilities, that allows a router to be brought down. And that creates large connectivity problems for the Internet. So, yeah. Anyway, all of us, every OS that does not yet have a hardened stack needs their stack hardened. So we will get fixes for all versions of Windows, servers and end-user versions, as soon as Microsoft has them. And we can hope that's as soon as they fix it, exactly.

Leo: Hurry up, guys. Very interesting stuff. If you want to read this in a transcript, I think it might be helpful to understanding it. I'm certainly going to do that. And Steve's got those transcripts online at his website, GRC.com. That's short for Gibson

Research Corporation. It's also where you find SpinRite, the world's best hard drive maintenance and recovery utility; all his great free security tools, too, like ShieldsUP!; some fun stuff like Wizmo. GRC.com. So do you want to do...

Steve: Let's just switch and do the Q&A next week.

Leo: That's what I was going to say, okay.

Steve: We'll just sort of slide ourselves back. I don't see any reason why we have to keep them on even and odd the way they have been.

Leo: Well, because I'm easily confused. But other than that.

Steve: No, see, Leo, the problem is, even and odd for Q&A, and you've got a three-advertiser rotation schedule. So that's constantly out of phase, and it's going to get to be a real problem.

Leo: if people want to ask questions, and I bet you there'll be a few about this, where do they go? SecurityNow.com, I mean, GRC.com/...

Steve: Feedback.

Leo: Feedback. I couldn't remember, either.

Steve: GRC.com/feedback. We also have newsgroups at GRC that are active, and I know we'll be discussing it there. I'm sure this will be something we're talking about in each Security Now! podcast for the next several weeks.

Leo: Until it's fixed, yeah.

Steve: As this issue moves forward because this is big news. This is not good.

Leo: Thank you so much, Steve Gibson. And I really appreciate your willingness to kind of say, wait a minute, this is something we've got to cover; do the research; and come back and be ready to talk about it. So thank you so much.

Steve: Well, I didn't want to change the show's name to Security Then!. I thought...

Leo: Now, right now.

Steve: Security Now!.

Leo: GRC.com. Also 16KB versions there. And we've started a new service at TWiT which of course could be brought down by bad guys at any time, I presume, since it is a server, although it's not a web server, it's a icecast server, which gives you a chance to listen to the audio wherever you are, even on an iPhone. Many phones support this. It's in beta test right now. I've applied for the name TWiT.am. But we do audio icecast streaming of the shows as we record them live, and then overnight we do repeats. So if you can't watch the video at live.TWiT.tv, you'll soon be able to listen to the audio at TWiT.am. There's details on my blog if you want to start doing it now during the beta test phase. And I've been listening on my iPhone, and it works great. It really is, even on Edge, you can go around town, drive around and listen. It's pretty cool. Pretty fun. Steve, thanks so much. We'll talk again next week, maybe with some good news.

Steve: Talk to you then, Leo. Bye.