Citrix Bleed

Description: What caused last week's connection interruption? Is it possible to create and maintain an Internet whitelist? What's the latest on LastPass vault decryptions? How do you know of a remote correspondent adds a new device to their Apple account that it's really them? Might there be more life left in Windows 10 than we thought? What's foremost in the minds of today's bug bounty hunters? What new free and open source utility has CISA released? Could it be that SpinRite 6.1 is finished? Is TLS 1.2 ready for retirement? And what about IPv4? How can open source projects get their code signed? And then we're going to take a really interesting deep dive into the Internet's latest mass-casualty disaster.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-946.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-946-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. He's got some really interesting stuff, including a new idea for zero trust network architectures. We'll also talk about more LastPass hacks, sad to say. And then he's going to do one of my favorite things, which is look at a major security flaw, this time with Citrix, and actually examine the code that made it possible. This is a real great learning episode for anybody who has to write code or just wants to know how these break-ins happen. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 946, recorded Halloween, that's Tuesday, October 31st, 2023: Citrix Bleed.

It's time for Security Now!, the show where we cover the latest security news, your privacy, your security, your online health and welfare with this guy right here, Mr. Steven Gibson of GRC.com. Happy Halloween, Steve.

Steve Gibson: Happy Halloween to you, you know, you are my namesake. You're the SQRL.

Leo: I thought I'd wear a SQRL outfit for you, yeah.

Steve: That's very appropriate. I have a SQRL right here.

Leo: Yes, you do.

Steve: In the camera on my whatever this thing is called that holds the microphone. Maybe it's a microphone holder.

Leo: Yeah, or "boom arm" we call it, yeah. And that's called a "mic flag," just in case anybody asks. You have a SQRL mic flag, yes.

Steve: I proudly do. Yes, the SQRL that took seven years of my life and will not give it back.

Leo: And now that I've shown you my SQRL head, please, children, do not fear, there is a human in here.

Steve: Oh. There's Leo, okay.

Leo: I won't take off the striped overalls, though. That's for another time.

Steve: No, stay in your prison garb, that's good.

Leo: What, my friend, are we discussing today?

Steve: Oh, have we got a good one. It is not often that we get a horrible problem in a - well, actually it was last week. Okay, but aside from that, we have another horrible problem...

Leo: Wow.

Steve: ...in a globally distributed, highly used piece of network appliance. But what is so cool about this is that we're going to get to do a deep dive into the exactly code mistake that was made in a way that all of our listeners, maybe not their significant others, but all of our listeners will be able to almost certainly understand...

Leo: I love it when you do that, by the way. That's so much fun.

Steve: ...and get something from it.

Leo: Yeah.

Steve: This is just - this one, it wasn't my plan. I was going to move forward with the NSA/CISA Top 10 misconfigurations for cybersecurity stuff. But when I realized that a research firm had done the reverse engineering, and that we'd be able to walk through that process and look at the code, and it's understandable, like the mistake that was made, it's just like, oh, this is just too good. So Security Now! Episode 946 for this Halloween, last day of October - and we're all going to be changing our clocks, those of

us who do that, on Sunday - is titled Citrix Bleed, reminiscent of the Heartbleed flaw in servers of the past. But we have a bunch of fun stuff to talk about.

We're going to look at, are going to answer the question, what caused last week's connection interruptions that we briefly suffered? Is it possible to create and maintain an Internet whitelist? What's the latest on LastPass vault decryptions? How do you know, when a remote correspondent adds a new device to their Apple account, if it's really them, you know, the people you think it is or not? Might there be more life left in Windows 10 than we have been led to believe? What's foremost in the minds of today's bug bounty hunters? What new free and open source utility has CISA just released? Could it be that SpinRite 6.1 is finished? Is TLS 1.2 ready for retirement? And what about IPv4? How can open source projects get their code signed? And then as I said, we also have a great Picture of the Week. But then we're going to take a really interesting deep dive into the Internet's latest mass casualty disaster.

Leo: Good lord. This is all in today's show.

Steve: Yes. And that's in the first five minutes.

Leo: Wow. Wow. You've jam-packed our Halloween episode with a lot of scary stuff. And that's I guess as it should be.

Steve: That's our plan.

Leo: I am ready. I have queued up the Picture of the Week. I don't understand this at all.

Steve: I captioned this picture "What's the plan here?" because what we have is some apparently new asphalt that has been paved around what looks like a phone pole and an electrical pole, which - okay. So what I know from the person, from the listener of ours who actually pulled off the road...

Leo: To take this picture?

Steve: ...in order to take this picture for us and send it because he thought, okay, we have to send this to Steve.

Leo: Picture of the Week for sure.

Steve: Yes. There's actually a new road that's been added off to the right. You can sort of see it toward the bottom of the picture heading off the picture to the right.

Leo: Ah, yes, yeah.

Steve: So the old, the original road is a - you can see it's got a yellow line down the middle, and it looks like the blacktop on it has turned gray because it's been around for a while. So that's the original road. Well, they decided to add a right turn lane onto the original road to complement turning to the right into this new road that's been added off to the right. Unfortunately, there was an existing two telephone poles. Actually a telephone...

Leo: Yeah, you can see in the distance the same configuration, but it's on the grass. It's not in the road.

Steve: Right, right. And this didn't used to be on the road, Leo.

Leo: No.

Steve: This used to also be on the grass.

Leo: Just like this, yeah.

Steve: But they decided to retract the grass and add some road.

Leo: Oh, boy.

Steve: But, you know, you can't drive through the phone poles.

Leo: No, bad idea, yes.

Steve: Because that would not be - that would not turn out well.

Leo: Yes.

Steve: No. So, now, I was thinking about this. They clearly have to move the phone poles back over onto what will soon be grass after they reseed this region. And moving them to the right would add slack to them because we can see that it's going to be - so that would kind of work, although there are some lines running off to the left. They're going to have to disconnect those and then stretch them or lengthen them and then...

Leo: What a mess.

Steve: You know, anyway, I just thought this was interesting because really how did this happen? How was it, I mean, people came along, and they added new concrete curbing to the right of the poles.

Leo: And expanded the shoulder.

Steve: Exactly.

Leo: Where the poles are.

Steve: Pulled that back, got everything ready, and put down new blacktop carefully around the poles so that it looks like they sprouted from the blacktop.

Leo: I'm just grateful they have two bright orange security pylons there to prevent people from driving into those poles.

Steve: That's right. You would not want someone to think, well, I'd better turn right here and get into the right turn land in order to do that.

Leo: Good lord.

Steve: Oh, goodness.

Leo: The number one reason our power goes out in this area is because people - it happens at least every few months - drive into a pole. They knock the pole over, power goes out, the electric company comes out and restrings the wires.

Steve: Those pesky poles. Now, you realize, Leo, that in more recent communities...

Leo: They're underground, yes.

Steve: The power is underground.

Leo: In fact, ours in our cul-de-sac, everything is underground, and I love it because there's no - you don't - this is so unsightly. Why are there two different poles right next to each other, too?

Steve: Well, one is power. The tall one is power. And the lower one is phone lines because you can see that black junction box there going away from the lower one is the telephone cable.

Leo: Ah, yes.

Steve: So, yes. For what it's worth, Leo, we have underground power here.

Leo: So much better.

Steve: And I can tell you that it is not always a lot better.

Leo: Oh, really. Go out a lot?

Steve: The reason I had my own generator was that the power here has been notoriously bad. Lately I think they finally changed the thing that was bad. It's a lot better. Anyway, I have two pieces of miscellany, one quick one, and then sort of something startling that I want to share. The first is why were we interrupted a couple times last week. After the connection interruptions which occurred during last week's podcast, I got serious about tracking down the cause of what, you know, I'd sort of been seeing them for a while, but they weren't bad enough to get in the way of the podcast. And I thought, okay, that's the last straw.

Leo: Yeah. We edited it out, but you froze onscreen.

Steve: Yeah. And I could see, because I'm monitoring my bandwidth on a separate screen over here, I could see a big red band twice come up where there was a lack of connectivity completely. So I decided to get focused on this. And the problem is normally I'm busy coding. And when I'm focused, I am a little OCD. I just sort of let everything else happen around me. I don't really pay attention.

Anyway, I was able to track down the cause of the trouble to occasional spontaneous crashes and reboots of my little Netgate SG-1100 router, which runs pfSense and sits just inboard of my cable modem. So it enforces my network segmentation, creates separate WiFi domains and does a bunch more things that I've talked about in the past you know, static port mapping and all kinds of cool stuff. I love pfSense. There's never been any - oh, even now it runs - it is a DynDNS client, and its firewall rules are adaptable, that is, the firewall rules will track any changes reflected by Dynamic DNS. So it allows my two locations that have public IPs to track each other for the sake of keeping them connected. So, for example, my Synologies are able to talk to each other, even though there's no public presence of the Synology sync going on. Anyway, it's very cool.

So what I wanted to mention was the first thing to check when something like this is happening is the device's power supply. Those little wall warts, as they're often called, that ship with consumer-grade equipment, are often the source of trouble. If the power supply is getting tired, it could be the reason for weird behavior. So fortunately that little Netgate SG-1100 uses a standard 12-volt DC supply that has the standard DC barrel connector. So I had plenty of swap-in replacements at hand. Ever since I exchanged its power supply, which was two days ago, I've had zero reboots. You know, I still have my fingers crossed. But since it was previous happening, sometimes several times a day, it really looks like the problem is solved.

So anyway, just a little note, a reminder that oftentimes the cheapest piece of equipment in the system, which is those little cheesy transformers these days, the little power supplies, they can have like much lower quality guts than the device they're powering, and they can bring it to its knees. So just a reminder about that.

Okay. Now, Leo, you'll remember another of this podcast's very early sponsors, the Nerds On Site guys, who are based in Canada, with their Nerds spread far and wide to help non-computer-savvy users and enterprises with their computer problems.

Leo: Yeah, I'll always have a soft spot in my heart for them.

Steve: I do, too.

Leo: Yeah.

Steve: You know? They were early fans of the podcast. And in fact I keynoted...

Leo: David Redekop, yeah, yeah.

Steve: Yes, Dave Redekop. I keynoted their annual gathering a couple times.

Leo: Oh, nice, nice.

Steve: And in fact I also met with them for the first time when I was in Toronto when I was serving as a guest on your Call For Help show which you were doing for Rogers Cable up in Toronto.

Leo: Yes.

Steve: So we're talking 18 years ago in those very early audio-only days of the podcast, when we used to record the show in your hotel room. So I remember you had a little recorder, and I would bring the microphones, and we'd set up and talk for like...

Leo: We tried once on the rooftop restaurant of the hotel. But that didn't work out so well.

Steve: No, no.

Leo: So we retreated to the room, yes.

Steve: Okay. Anyway, since then I've stayed loosely in touch with David Redekop, who was one of the principals back then, and I've sort of been vaguely aware of what he's been doing. Well, he was passing through town last Friday, so we rendezvoused at Starbucks to catch up, he and two of his guys. Okay, now, this was over my second five-shot latte of the morning, so my brain was fully charged up. David took about an hour to walk me step-by-step through the careful and deliberate design of the network security solution he and his team of techies at ADAMnetworks have designed, and which is currently in use actively protecting around two million user and device endpoints around the world. So this solution is not just theory, and I think a surprisingly good idea. It's actually in place and working.

And once I'd heard what this group has managed to pull off, that didn't surprise me because it felt like the result of a decade or more of iterating on a concept in order to bring it to maturity. You know, you start with an idea, you implement a first pass, and then stick it out there. You remain involved and connected. You learn what does and doesn't work. Then you iterate and push out an improved solution. You add new features to address new requirements. You test them and adjust them until they're working correctly.

Now, I've often noted on this podcast that I cannot imagine the challenge of keeping an enterprise safe while it's connected to today's Internet. And in fact, today's Citrix Bleed topic makes me shudder a bit because, even if everything is done right, it's still possible to be taken down. I'm so glad that keeping a highly connected sprawling enterprise safe is not my job. But it does still need to be done, and these ADAMnetworks guys have not only imagined the challenge of keeping an enterprise safe, from what was explained to me on Friday - and I think I pretty much understand the way it works, at least in broad strokes - they've not only imagined the challenge, they've risen to it. And if I had an enterprise network which I needed to protect today, based upon what I understand now, I don't think I'd want to be without this solution. I'm thinking of reaching out to Father Robert to make him aware of this because it would be so cool to have the Vatican this well protected.

Okay. So what did they do? Nobody today runs their firewall where by default everything is allowed except for traffic that's known to be malicious. In the dim dark ages of the Internet, once upon a time we once blocked known exploitable ports. We don't do that anymore. But blocking known bad domains or scanning communications traffic for known malicious content is what's being done today, and it's the modern equivalent of a default "allow all" rule at the bottom of a list of things that you want to block. We do it because we think there's no alternative. Last Friday, the ADAMnetworks guys showed me a better way.

What these guys have done is to not worry about inspecting and interpreting the contents of enterprise user communications, or anyone's communications. As I understand it, they do not do any deep packet inspection so no one's communications contents privacy is ever compromised. What they've focused on and specialized in and made work is only allowing connections to known safe and benign entities. Now, okay, if it were possible to do this perfectly, phishing attacks would be ineffective because a user clicking on a phishing link would be unable to obtain any malicious content. And even if someone were to inadvertently bring some malware into the enterprise that infected it from the inside, if that malware was then unable to connect to its command-and-control servers, it would be effectively inert.

So imagine that you start with very strong DNS filtering and require all clients on the network to use this DNS. They've also extended this through and enforced it for enterprise-controlled smartphone and other mobile clients so you get complete coverage. The thing that sets this apart from other solutions is that this is not the typical blacklisting DNS, which selectively blocks known bad domains. ADAMnetworks has managed to build a practical whitelisting DNS which by default blocks everything unknown and only permits DNS resolution for known safe domains. DNS is resolved on premises, so privacy is preserved; but it's also informed by a shared central manager in the cloud so that as new domains come online and are cleared, the master whitelist can be updated.

If a user attempts to connect to an unlisted DNS domain, they receive an intercept page. And the way this is done is quite clever. I won't go into it, but it works. And they must manually accept and approve that if they want to proceed. But before that is allowed, 18 other public sources of DNS filtering from other providers are consulted to make sure the unknown domain is not known to be malicious by anyone. And there's also technology in

the pop-up intercept page which prevents malware from being able to simulate the human giving their permission.

And speaking of malware, it turns out that modern malware is hip to having its DNS blocked. So there is malware out there that has static IP addresses burned-in. They don't do any DNS. It just connects directly to its command and control. So this introduces the outbound firewall component of this solution which they call "Don't Talk to Strangers." No outbound connection is allowed to any IP that wasn't first looked up with the system's local DNS resolver. So by linking DNS resolution to a dynamic outbound firewall...

Leo: Clever.

Steve: Yes. DNS, which has already been made very strong, becomes the universal gatekeeper. Okay. But then it turns out that Facebook, for example, has some apps that also don't do DNS. Facebook's apps are benign, and they know the fixed IP addresses they want to connect back to at the mothership. So to make a solution work completely, which they have, it's necessary for the system to incorporate some whitelisting for known benign blocks of destination IPs. So that's in there, too.

Okay, now, what I've just described is an overview which just scratches the surface of the enterprise security solution these guys have developed. And I'm sure that our techie listeners are saying "but but but but but," to which all I can say is that they actually have this thing working. They demonstrated it to me; and, as I said, more than two million user and device endpoints are currently under this solution's protection.

They have a full mature-looking management UI, with scheduling and per-user permissions and everything else needed to implement a practical enterprise-class solution. I asked David how much trouble there was with unknown, yet benign, DNS; and he acknowledged that it was there, but that they had reduced the level to such a degree that it was effectively negligible. I recall that he said 5%, but I don't remember what 5% was of.

So my reaction was that, if I were an IT guy, like with this kind of responsibility, and it was possible and practical to operate behind what is essentially a whitelisting Internet firewall, if that only caused a tiny bit of trouble in fringe cases which can then be worked around, that would be entirely acceptable for my peace of mind and in return for the enhanced security it would provide for everyone else, all of the time. And what's cool is that it protects against the unknown.

It turns out that the widespread SolarWinds vulnerability which took the whole world by surprise would not have affected any of ADAMnetworks' users. And the always evolving Pegasus smartphone spyware would not function if it were to infect any of the handsets protected by this system. So as usual, my focus and interest was on the technology, which I think is clearly pretty cool, and which I felt an obligation to share because this could help a lot of our listeners. Although I cannot vouch for this from firsthand experience because I have none, you know, we do know these guys. And we've known them for nearly 20 years. They're techies like us.

Just take a look at any of the many videos posted on their website and how excited they are about what they've created, and you'll see that immediately. I was very impressed by what I saw last Friday. I don't know anything about the business side of this, you know, like what it costs or what plans they have or any of that. We didn't talk about that. We were talking about the tech. My sense is that the solution is scalable down to a single router in a small office all the way up to a global enterprise because there was - he did make some mention of, like, the massive amounts of traffic that they're able to handle.

But I don't know what the extreme points are for that. As I said, I also have no idea what it costs. You'll need to track that down for yourselves, if it sounds interesting.

So anyway, after what I heard and learned Friday, I wanted to put this on everyone's radar. They are ADAMnetworks. It's A-D-A-M-N-E-T dot works. And I have some links in today's show notes for their site, although I got the links just by poking around a bit the other day. So you'll find a page of videos. Even their wide-open tech support portal so you can see everything that their customers are asking and, like, being confused by or needing help with. So anyway, I just wanted to share this because I was really impressed that they could make this work. And boy, you know having a default deny, allow trusted system that could protect something the size of an enterprise, to me that's a game changer. So I will leave it up to our listeners to decide if it looks like something for them.

Just last week, the still unknown hackers who breached LastPass and exfiltrated everyone's encrypted vaults managed to siphon off another $4.4 million worth of crypto-assets from 25 of LastPass's previous and maybe still current customers, those who haven't run for the hills. We know that most LastPass users will have remained with LastPass and won't be receiving the news that stolen vaults are being actively decrypted. As we saw previously, the hackers are cracking the stolen LastPass password vaults, presumably those that were protected by weak passwords and probably also few PBKDF iterations, though I would argue by no fault of the user. That was squarely on LastPass for not bringing users up to speed. That is, their password iteration counts to protect against having weak passwords, which we know users are going to have.

Anyway, from those vaults that they decrypted, they recovered the crypto wallet seed phrases and then drained those customers' user accounts. So with the latest round of thefts, which are being tracked because we know the wallets into which those stolen funds are being transferred, this brings the total to nearly $40 million after they had previously stolen around 35 from about 150 other users earlier this year. So it's ongoing, and it's really sad and unfortunate.

Leo: And if you had a LastPass vault with your keys in it, you might want to change your keys.

Steve: Yes, yes. And in fact remember that we read a note from one of our listeners who had set up another wallet and was planning to transfer his currency to it, but didn't get around to it.

Leo: Yikes.

Steve: And lost three something million dollars, I think it was.

Leo: Yes, yes. If you don't have around to it, you'd better get around to it.

Steve: That's right.

Leo: All I can say.

Steve: Remember that cool feature of Threema that I always liked so much? It was the ability to positively verify the other person's public key through some out-of-band communication channel. You know, you can't verify it in-band because if you're not talking to the person you think you are, then you're still not talking to the person you think you are after you exchange key verification somehow. So it's got to be like some other non-in-band communication app channel, like a phone call or, even better yet, a face-to-face meeting.

Okay. So Apple hasn't gone quite that far since they continue to manage all of their users' keys on their behalf, which I'm sure is a good thing in the case of your typical Apple user. But they have developed a slick new way to verify any new device that registers itself with a user's account. They call it "iMessage Contact Key Verification," and it has just appeared in the developer previews of iOS 17.2, macOS 14.2, and watchOS 10.2. It's designed to present an alert inside iMessage conversations whenever a new device is added to a participant's account.

Okay. So say that I'm in an iMessage dialog with someone, and they add a new device to their account and, you know, talk to me from it. How do I know that the new device was actually added by the person I think I'm messaging with? If it was someone else holding the new device and impersonating this person, that's not good. So this new system displays a fixed eight-digit code in both my phone and their phone, that is, the newly added phone. This allows us to arrange a telephone conversation or a FaceTime chat or something not just iMessage, where I have some absolute knowledge that the person I think I'm messaging with is this other person. We compare eight-digit codes. That is, what I would do is I would tell the person, you know, holding the new device, okay, what eight-digit code is your phone showing you? And, you know, I know who I'm talking to; right? I mean, so because it's a telephone conversation or a FaceTime call, they read the eight-digit code, I confirm that that's the code my phone is sharing with me, and then I click on Mark as Verified, which then verifies that this new - it tells iMessage that I am accepting my remote correspondent's new device for subsequent iMessage communications.

So anyway, Apple's onscreen instructions say: "Verify who you are messaging with by comparing contact verification codes in person or over the phone." And at first I thought, wait, you know, shouldn't I make them tell me? And I guess that's the case; right? Except that, you know, if it's - basically what this does is it forces you to have a non-iMessage, some non-iMessage contact with the person whose phone is also showing this code. So you know that that's the device which you don't yet want to fully trust. So I'm not sure if it matters if I tell them the code, or they tell me, or we alternate digits back and forth, or they tell me the first half, and I tell them the second half? I don't - I'm not quite sure.

Leo: We leave this as an exercise. You figure it out.

Steve: But it does force you to have a face-to-face confirmation. And then you say, okay, fine, you're holding the phone, you told me the code, that's good. I'm now going to accept this as belonging to you. So, yeah. Anyway, cool that Apple continues to do this. They're not perfect, they make mistakes, I get that. But, boy, you know, every contact we have with the measures that they are taking seem really proactive. And, you know, they've got people sitting around just thinking of how to make our stuff more secure.

Leo: This solves the problem of somebody rekeying or getting a new phone in this case. And Signal has done it, as you said, Threema does it. How do you verify that

this new communication with a new key is the same as the old person? And it just shows that Apple's now doing strong encryption on messaging; right?

Steve: Right, right. Well, stronger authentication.

Leo: Actually it doesn't say anything about encryption, does it. It says [crosstalk].

Steve: Right, yeah. And we know that they always had really good encryption. On the other hand, if you're not who's decrypting it at the other end, that's not good either.

Leo: Right, right. Authentication's a big part of this, yeah.

Steve: Right, exactly. And of course that's the whole certificate thing; right? I mean, anybody can have a certificate that brings up an encrypted connection. What you want is to know who has that certificate and for their identity to have been proven.

Leo: Right.

Steve: So and that's what Let's Encrypt does not try to do. It says it's only saying the domain that asked for this certificate was granted one. So now nobody can see what's going on in private, but it's not making any assertations, I mean, it's not making any assertions about who it was that it issued the certificate to. That's why, you know, I'm still using a fancy certificate from DigiCert where I have to jump through hoops every so often in order to say, yup, still me. Hi, Mom.

Leo: Yeah.

Steve: Okay. So perhaps there's hope for longer Windows 10 support. Remember I just talked about last week how like in the week before we had just crossed the less than two years remaining for Windows 10 line, which to my mind seems really premature.

Okay. It turns out that Windows 10, Leo, is good for the Earth. Or more correctly, it's that Windows 11 is less so. More than 20,000 members of the Public Interest Research Group (PIRG) have formally asked Microsoft to extend support for Windows 10, which, as I recently mourned, is scheduled to reach end-of-life status in two years, in October of 2025.

Leo: It has been 10 years, or will have been. I mean, that's a long time.

Steve: But it still feels new. It still feels new to me.

Leo: Yeah, as you used Windows 7 until last month.

Steve: I'm still in front of Windows 7; that's right. And my browsers are not happy with me. But okay. The organization, this PIRG group, says that, get this, around 40% of the one billion devices that run Windows 10 cannot be upgraded to Windows 11.

Leo: Wow.

Steve: And that will create an unnecessary e-waste problem.

Leo: 400 million, to be precise.

Steve: Yeah, uh-huh. Of course, on the flipside, Microsoft and many other hardware vendors are looking forward to this event for just that very reason. PIRG is the same group, the same industry group that had recently convinced Google to extend the warranty and support for older Chromebooks, using similar arguments. It's like, don't put these out to pasture. It's going to create a bunch of toxic landfill, and what's wrong with them? They still work. They're still just fine. Just like Windows 10, Microsoft.

Leo: Or it's just good for Linux. I mean, honestly, just put Linux on the damn things, and you'll never have to worry about Microsoft again.

Steve: Right. You will have to pay people to teach you how to use it, however.

Leo: Oh, yeah. [Indiscernible] it's the same as Windows. Easy-peasy.

Steve: So it is certainly somewhat bracing to learn that 40% of current machines, a billion current machines, won't be compatible with Windows 11. I mean, we've been talking about this from the beginning, and we know that it's just because they don't want it to be. Right?

Leo: Yeah, right, right. They have to have 8th generation Intel processor, they have to have TPM 2.0, even though that that's not necessary for the operation of Windows 11.

Steve: Right. It is legitimately...

Leo: But honestly, Linux works great on that machine. I'm just telling you.

Steve: It is, yeah, the problem is that users don't. It's legitimately a huge deal. So I really do wonder whether Microsoft, you know, is prepared for the user and the corporate ire that will follow from being told, sorry...

Leo: They've been warned. It happens to every version of Windows. And 10 years is, as you point out, much longer than the Chromebook. It's much longer than any phone you've got. Google just got a lot of attention for saying we're going to support

this phone for seven years. So I don't think, I mean, look. Look at your phone. Most people get new phones every few years. I mean, the landfills are jammed with stuff. What about your new car? How long does your car last? And that's a lot of physical waste. Many computers' worth. We've got to do something about this. You can't put Linux in your car, unfortunately.

Steve: Why force the hard work to be obsoleted for Windows 11, which does not require, we know it doesn't require that the hardware...

Leo: No, I agree. I agree. They shouldn't say 8th generation and TPM 2. But increasingly they've added features that do support, maybe even require that newer hardware. But I think you nailed it when it's really designed to support the PC industry and people buying new PCs. I mean, that's clear. But the whole - our whole industry is geared around getting people to throw out their old stuff and buy new stuff.

Steve: So last Thursday, the HackerOne bug bounty folks...

Leo: Oh, yeah. We saw this. Yeah, really good stuff.

Steve: Yeah, neat stuff. They're neat people. Took the occasion of their total bounty payouts crossing the $300 million mark to provide some insights into the current state of their operations. I've edited it down a bit, but there's some interesting stuff in here. So just to be clear in the following text, when they refer to their "customers," they mean organizations who pay bounties to have their bugs found in their own code. And of course when they refer to "hackers," those are the freelance agents who are the finder of those bugs.

So HackerOne said: "HackerOne today announced its ethical hacker community has surpassed 300 million in total all-time rewards on the HackerOne platform. And 30 hackers have also earned more than $1 million on the platform, with one hacker surpassing $4 million in total earnings." So, yeah, it's possible to have a career.

"And hackers are finding new opportunities to earn more by diversifying their skill sets as emerging technology reshapes the threat landscape. 55% of hackers surveyed" - and they surveyed 2,000 of their hackers - "expect that Generative AI will become a top target in the future. Crypto and blockchain organizations continue to see strong program engagement, offering the highest average overall rewards for hackers, including the award of this year's top payout of $100,050. HackerOne's customers have also expanded how they commission hackers outside of traditional bug bounties with pentesting engagements increasing by 54% on the platform this year in 2023."

So, and they have some bullet points. Hackers continue to experiment with Generative AI, as 61% of hackers said they will use and develop hacking tools from GenAI to find more vulnerabilities, and another 62% of hackers plan to specialize in the OWASP Top 10 for Large Language Models. Hackers also said they plan to use GenAI to write better reports, 66% said that, or code, 53% said that, and reduce language barriers. That was a third of the people.

Hackers report insufficient in-house talent, meaning on the part of HackerOne's customers. Hackers report insufficient in-house talent and expertise as the top challenge for organizations, which is a gap they are filling. 70% of customers stated that hacker

efforts have helped them avoid a significant cyber event. 57% percent of HackerOne customers believe exploited vulnerabilities are the greatest threat to their organizations, over phishing at 22%, insider threats at 12%, and nation-state actors at 10%.

Customers are getting faster at fixing vulnerabilities, as the average platform-wide remediation time dropped by 10 days in 2022. Okay, now, I didn't see the report because I had to go through a bunch of hoops to get it, and it didn't seem necessary. But I would like to know dropped from what to what by 10 days? You know, did it drop from 20 days to 10, in which case it was in half, or from 100 to 90? To me it makes a difference. But anyway, their summary didn't say. They said, interestingly, automotive, media/entertainment, and government verticals saw the biggest decrease in time to remediation, with an over 50% improvement. Okay, so those industry groups - automotive, media/entertainment, and government - were in this most recent update much faster to jump on trouble and fix them.

And finally, organizations are reducing costs by embracing human-centered security testing earlier in their software development lifecycles, with customers saving an estimated $18,000 from security experts reviewing their code before its released rather than afterwards.

So Chris Evans, HackerOne's CISO and Chief Hacking Officer, was quoted saying: "Organizations are under pressure to adopt Generative AI to stay ahead of competitors, which in turn is transforming the threat landscape. If you want to remain proactive about new threats, you need to learn from the experts in the trenches: hackers."

So anyway, that annual Hacker-Powered Security Report is based on data from HackerOne's vulnerability database and gathered the views from both HackerOne customers and more than 2,000 hackers who were interviewed on the platform. And to me, it seems that I guess it's not surprising, but maybe only in how quick it's happened, an interesting takeaway that mistakes are bound to be made in the rush to get new Generative AI solutions rolled out and in the world, which did seem to happen pretty quickly. So getting up to speed on AI technology appears to be on everyone's radar. No doubt the bad guys and also the white hat hacker good guys who are going to help companies to solve their problems before they get exploited.

Logging Made Easy, from your friends at CISA. Last Friday, our own CISA released a tool called Logging Made Easy through their official GitHub account. LME is a toolkit that enhances log management on Windows devices. The tool was originally developed by the UK's NCSC agency way back in the late 2010s, and they retired it in March of this year. So CISA picked the tool up, updated and re-wrote it to cover recent Windows versions and to enhance its logging capabilities.

Here's what CISA said. They said: "Today, CISA announces the launch of a new version of Logging Made Easy (LME), a straightforward log management solution for Windows-based devices that can be downloaded and self-installed for free. CISA's version reimagines technology developed by the United Kingdom's National Cyber Security Centre (NCSC), making it available to a wider audience. Log management makes systems more secure. Until now, it's been a heavy lift for many targeted organizations, especially those with limited resources. CISA's LME is a turnkey solution for public and private organizations seeking to strengthen their cybersecurity while reducing their log management burden. LME builds upon the success of the NCSC's log management solution, and CISA urges organizations to secure their Windows-based devices today by downloading the free LME technical solution."

Over on their GitHub page, they said that Logging Made Easy can show where administrative commands are being run on enrolled devices, see who is using which machine. In conjunction with threat reports, it's possible to query for the presence of an

attacker in the form of Tactics, Techniques, and Procedures. So anyway, all that seems pretty cool and useful. The link to jump there is this week's GRC shortcut, so grc.sc/946. But the full link is also not very difficult, it's github.com/cisagov/LME. And that'll take you to a place where you can find out more and grab it for yourself.

And Leo, I have here in the show notes, under the heading of SpinRite, the issues page...

Leo: I love this. Yeah.

Steve: ...from GitHub, from our GitLab, as of Sunday. And I checked a few minutes ago. It is still holding. It says: "There are no open issues." And then further up on the page it says, you know, Steve Gibson, I'm the project author, SpinRite v6.1 issues. Open: 0. Closed: 536.

Leo: That's impressive. That is really impressive.

Steve: And of course those are all the things that we tripped over and stumbled upon. Some bugs in SpinRite. Many bizarro behavior in weird things that caused me to go to eBay and buy some old motherboard, and for Lorrie to say, uh, we're going to put these away someday; right? Uh, yes, dear.

Leo: Are they on the dining room table? I mean...

Steve: They are out in the open.

Leo: Did you run out of room in your office?

Steve: For all and sundry to see. So...

Leo: Oh, lord. Poor Lorrie. Hey, she knew what she was getting into.

Steve: So on Sunday - she does and she's still supporting my work. So on Sunday SpinRite's second and quite possibly final pre-release was published. As far as I know at this point, the DOS executable code that's now available will be what finally ships.

Leo: Woohoo!

Steve: Yeah. SpinRite 6.1 includes a tiny DOS text editor for viewing and editing its log files from DOS, and to make editing any DOS config files easier. That editor also appears to be completely ready. I need to verify that all of the FreeDOS OS kernel customizations that I have made for SpinRite are complete. You know, I needed to tweak the FreeDOS kernel so that it doesn't completely go crazy if it sees a GPT partition because all it knows about is MBR format. And also so that it doesn't freak out if there's like errors on the disk that it's trying to log into in order to create DOS-style drive letters for all of the drives the way we used to in the old days.

So I've done all that. I think it's done. I'm just going to go back and check because it's been a while. Then I will return to finish a bit of the work on the Windows side of SpinRite, which is the thing that creates the boot media for SpinRite. And basically I'm going to incorporate the improvements in that technology that arose from the work that I did on ValiDrive. So I'll port those back over into the Windows side of SpinRite. I need to update GRC's servers to digitally sign individually licensed copies of SpinRite on the fly for customer download. And at that point, SpinRite 6.1 will be available as a finished product to all SpinRite 6 owners, and then I'm going to begin work on SpinRite's web pages to document all of its new operations and screens to show all of the many changes and improvements that have been made during the past three years of this project.

And then, after I get an email system in place to begin notifying SpinRite 6.0's 19 years' worth of existing users, I get to start in on SpinRite 7, which really excites me. But in the meantime, we really do have a major step forward here. This morning I saw a posting in GRC's web forums which said: "Thanks to the pre-release of 6.1, I'm running SpinRite on drives that I either have not been able to run it on for years, or drives that were just too big for 6.0, so I was never able to run SpinRite on them before." So, you know, promise made and promise kept.

Leo: Whoo, boy, that's exciting.

Steve: And quite soon, promise delivered.

Leo: Yay.

Steve: Yeah, I'm very happy. So we have a bit of closing-the-loop feedback. Shep Poor said: "Hi, Steve. You've trained us well. When I saw a BleepingComputer article titled 'Windows 11 adds support for 11 file archive formats,' my first thought was, 'Oh, great, 11 new interpreters built into Windows, ready to be exploited.'"

Yes, Shep. I appreciated his tweet because "thinking security" is probably the best thing our listeners could take away from the many lessons we learn here every week from specific events. Those events will come and go. But seeing the underlying issues and what connects them together, you know, what are the takeaways and the lessons, those will endure and remain viable and valuable long after the Unix epoch has shut down everything that we know and love.

Leo: It's true. It's in my head. I think the same thing. Oh, another interpreter.

Steve: Yup.

Leo: And it's because of you. I mean, I totally think of these security flaws because you've trained us all these years.

Steve: Because we've looked at so many. And boy, are we going to look at a doozy here in a minute. Someone named BP, whose handle is @Wildlandsguy, he said: "Hi, Mr. Gibson. I had a huge smile on my face when you announced you'll be doing SN past 999. I would appreciate hearing your opinion on what it would take to get the world to move

to IPv6. What are the significant hurdles?" And he said: "FYI, ran ValiDrive on three PNY drives from Best Buy, and all checked out okay." So, cool.

Okay. So there are probably still some backwaters of the Internet which IPv6 has not reached. But from everything I'm seeing, nearly everything appears to be IPv6 ready. So I think that at this point the only thing that's holding things back is a concern over breakage. What might break if IPv4 was withdrawn, and we were left only with IPv6? We've seen examples where surprising things have broken with change. We were just talking about the surprise that the designers of the upgraded TLS 1.3 protocol faced when it initially didn't work because too many boxes somewhere along the way in the traffic path broke it. They shouldn't have broken it, but they did. So it was necessary for the TLS 1.3 guys to make 1.3 appear to be much more like 1.2. Again, shouldn't have had to do it, but they did.

And we've seen how incredibly cautious the browser designers are whenever they change anything. They'll initially have some newfangled feature disabled and only enabled manually by intrepid developers. Then they'll turn it on by default for 1% of the population while closely monitoring that 1% for any trouble. Then they'll gradually start rolling it out, enabling it more and more, still kind of holding their breath and, like, making sure that nothing really horrible happens. I mean, so, you know, they learned to be super cautious from experience. So I think that's where we are.

In the case of IPv6, I think we're at the "if it's not broke, don't fix it" stage. The good news is that everyone is probably ready for it when it's forced upon us. But until it's truly necessary for someone not to have IPv4, until there just isn't any IPv4 left, and the only thing that newcomers can have is an IPv6 address, that's when I suspect IPv6 will finally begin in earnest, and really no sooner, until there's really no other choice because that's just the way all this stuff is, is like, you know, leave it alone, unless and until we have to finally give it up.

Someone tweeting as @NotDorion said: "Hi, Steve. I've recently seen the high cost of signing certificates, especially for open-source projects like ImageMagick," and he provides a link to ImageMagick's pages over on GitHub. He said: "On one hand, I see that having costly certificates makes it difficult for malware to access them. But maybe a solution is needed for open-source projects. Would be glad to hear your thoughts on this. Thanks for all the great content all these years."

So that was interesting. The posting on GitHub that our listener linked to was titled "The Windows installer for ImageMagick will no longer be signed." And its developer wrote: "Today, our code-signing certificate will expire. For many years LeaderSSL sponsored us with a code-signing certificate, but they're no longer able to do so. Since June of 2023 the CA/B Forum requires that OV code-signing private keys be stored on a FIPS 140-2 Level 2 or Common Criteria Level EAL4+ certified device." Meaning some sort of HSL, some hardware security layer. Some, you know, like we've talked about before, inside a truly secure enclave device.

He writes: "This means we are no longer able to export our code-signing certificate with its private key and use this in GitHub actions. We would now either need to have our own GitHub agent and hardware token or use a cloud solution," he says, "e.g., DigiCert. Our preference would be to use a cloud solution that integrates with GitHub. DigiCert seems to be our only option now, but a certificate there would cost $629, tax excluded, for a single year. If your organization requires a signed installer, then please consider sponsoring us with a code-signing certificate. Please reach out to @" and then his handle "for questions or in case of a sponsorship."

Now, what was neat was that in a terrific example of community action, this posting from three days ago generated a terrific thread which, based upon this developer's reaction,

appeared to provide a number of useful alternative solutions, one being a free solution for a year, and others involving Azure Code Signing, which appears to be a good thing. I didn't look into it any deeper. But other open source projects are having similar difficulties, and they chimed into the thread, as well.

So I wanted to share the news of this in the event that the eventual solution found by ImageMagick's developer might be of some use to some of our listeners. As we know, all of GRC's code is signed by a DigiCert EV certificate which works for me since I'm a commercial entity, and it's worthwhile. And I was just a couple weeks ago talking about, you know, how it matters to have your code signed. You're looking at downloading something. You want to make sure that that EXE hasn't been modified since the time it was signed by the entity that claims to do so. And that requires a signature.

The problem, of course, is that these signatures, the code signing, the private key in these code-signing certificates must be kept secure because you don't want bad guys signing stuff, claiming to be ImageMagick, and getting the trust that ImageMagick has been able to accrue over the years. So, I mean, the problem is the way we're going, the nature of open source software historically is kind of coming into collision with the fact that we really want to know, we want to know who's behind it.

Matt Davis tweeted and offered some great feedback about last week's look at privilege. He said: "Hi, Steve. Thrilled to hear you're continuing past 999. I've listened since the beginning, and I'm looking forward to another 999 or so afterwards." Okay, well, let's hope. He said: "You may have overlooked the heart of the Privileged Accounts issue you discussed during last week's episode. The problem is not lazy IT folks signing in as root in the morning. It's that we are trying to create a shift away from 1-to-1 parity between Accounts and People.

"Many moons ago," he wrote, "shared logins were the norm for system management, which has obvious downsides and drawbacks we all understand. Single-sign-on solutions like Okta aimed to solve this by promoting unique accounts per user. Each person gets exactly enough privilege for their needed tasks, and staff turnover is easy to handle by revoking a single password. However, this led to a new problem. Administrators only have a single account, which they have no choice but to use, even when they're performing basic tasks, and it's all powerful.

"The obvious solution would be to create separate Standard and Admin accounts for every administrator; but since most software is now is charged per account per month, this tends to be cost prohibitive." And he said, "We have one service at $300 per month per user." Yikes. He said: "Ideas like Just Enough Access help, allowing admins to switch between regular and elevated access, akin to UAC in specific environments like Azure and AWS. But slow adoption in userland means that I still need to use my superuser-level login to access my email every morning.

"Looking forward to having the option to submit via email. It seems like the Feedback Form hasn't really been working over the past few years, and you are the only reason I've pulled my Twitter account out of the dustbin. Love the show and SpinRite. Thank you."

So anyway, Matt, thank you for some terrific perspective from the field. And yikes, charging an organization $300 per user per month for authentication, that makes my blood boil a little bit. What a rip-off. I can't see what great deal of work they're having to do that justifies that much expense.

Okay. Finally, JP McNeal said: "Hi, Steve. I'm a huge fan of SN and so glad you'll keep going after Episode 999. I'm wondering what your thoughts are on turning off TLS 1.2? I did a little research and found it interesting that there's no official EOL [end of life] for

TLS versions. It's just a matter of when the big browser vendors decide to not support it. Microsoft, Apple, and Google agreed to stop supporting TLS 1.0 in 2018, for example. Thank you. Signed JP." And he said: "PS: I work for a hosting company." So, yeah, he's got a vested interested in, like, when will 1.2 go away.

So I suppose that time flies, but my impression is that TLS is still a relatively recent arrival, and that 1.3 hasn't fully happened yet. And I guess, you know, I think that about Windows 10 also. So what do I know? TLS 1.0 and 1.1 were both deprecated only two years ago in 2021. TLS 1.2 has been in use since 2008, and TLS 1.3 only began 10 years later and five years ago in 2018. But here's the stat that most matters: As of last month, 99.9% of all websites support TLS 1.2, but only 64.8% support 1.3.

Leo: There you go. Yup.

Steve: Yup. So not even two-thirds of the Internet's websites today are TLS 1.3 capable. And given a good choice of cipher suites for TLS 1.2, there's really nothing wrong with it.

Leo: That's an important point.

Steve: It can be made secure by choosing your cipher suites carefully. And so browsers could certainly choose not to support cipher suites that they felt were a problem and leave the cipher suites for 1.2 and the 1.2 protocol which are perfectly strong, continuing to be used. So the Internet as a whole is not going to be able to move beyond 1.2 for a while yet, not until adoption of 1.3 becomes far stronger than it currently is. And as we know, that'll just take time. New web servers will all be supporting 1.3, but older servers which only support 1.2 will need to either be upgraded or die, and then be replaced by new servers. So for a while it'll be 1.2 and 1.3, moving toward 1.3 over time.

Leo: Very closely related to the IPv6 question. It's interesting.

Steve: Yes, exactly.

Leo: People want us to move along, don't they.

Steve: Exactly. So our final sponsor. And then, oh boy, do I have some fun to share.

Leo: Oh, good. By the way, let me echo everybody's pleasure that you're sticking around past 999. That's good news for all of us. All right, Steve. You have promised...

Steve: Oh, boy.

Leo: ...to blow the socks, so to speak...

Steve: Leo, you're going to love it.

Leo: All right.

Steve: So as I said at the top of the show, I decided to punt our continuing exploration of the joint NSA/CISA Top 10 cybersecurity misconfigurations until at least next week, due to another mass casualty cybersecurity event that gives us a really interesting opportunity to take our podcast listeners on a journey into the interior code guts of a widely used and hugely popular piece of networking hardware whose recent wide-scale compromise is believed to have negatively impacted the network security of more than 20,000 enterprises worldwide who have been depending upon the integrity of this device to keep them protected. So this week we're going to take a deep dive into the specific code flaw that resulted in the coining of the term Citrix Bleed, which is also the title of today's podcast.

I've often spoken of the now common practice of jumping on and quickly reverse engineering vulnerability-closing patches to update code as a means of discovering the details of the defect that the patch corrects for the purpose of either, A, if you're a white hat security research firm, documenting an interesting piece of Internet history; or, B, if you're a black hat criminal organization using the information gleaned from such reverse engineering to begin compromising the hapless users of not-yet-patched devices.

As always, it's one thing to talk about this being done, and an entirely different thing to obtain a good sense of the details from someone who's actually doing it. So I smiled when I encountered a write-up from a firm wearing a white hat of the process they underwent to reverse engineer a really, really bad vulnerability that is, unfortunately, being exploited and is damaging people right this minute. Before I get into the nitty-gritty, I want to set the stage by sharing an overview of the situation written by security researcher Kevin Beaumont, whom we often encounter in the security space. He tweets as "Gossi the Dog" is his handle. I don't know why.

Leo: Okay. Very reliable source, yes.

Steve: That's right. Here's how Kevin described the situation three days ago. Under his headline "Mass exploitation of Citrix Bleed vulnerability, including a ransomware group," he wrote: "Three days ago, Assetnote posted an excellent write-up about Citrix Bleed aka CVE-2023-4966 in Citrix NetScaler. This vulnerability is now under mass exploitation. A few weeks ago it was under limited targeted exploitation to allow network access. It's not Assetnote's fault. It was clear multiple groups had already obtained the technical details. The patch became available on October 10th." Okay, so that's exactly three weeks ago today.

He says: "Even if you applied the patch and rebooted, you still have a problem as session tokens persist. The vulnerability allows memory access. Sounds boring; right? The same memory contains session tokens, which an attacker can easily extract. Those session tokens allow the bypass of needed login credentials and the bypass of all multi-factor authentication. An attacker can just replay the session key, and they're in. Who uses Citrix NetScaler anyway?" he asks. "Many tens of thousands of businesses run it. It is very, very common in enterprises and governments. If you think nobody runs this stuff, you probably also think everybody uses Linux on their laptop." That's what he said, Leo. I didn't say that.

He said: "For an additional perspective, the Risky Business Newsletter wrote the following. They said: 'A Citrix vulnerability has entered the dangerous stage of mass exploitation as multiple threat actors are compromising unpatched devices all over the

Internet in a race with each other to steal their session tokens. Known as Citrix Bleed and tracked as CVE-2023-4966, the vulnerability impacts Citrix ADC and Citrix NetScaler, which are extremely complex networking devices used in large enterprise and government networks in multiple roles, such as gateways, proxies, caching, VPN servers, and a bunch of other stuff.

"The vulnerability allows threat actors to send junk data to the Citrix OpenID component that will crash and leak a part of the device's memory. The bad part is that, in some cases, this memory may contain session tokens that attackers can collect and then bypass authentication to access the device and the network behind. Citrix released patches to fix the Citrix Bleed memory leak earlier this month, on October 10th.

"The bug in itself is extremely bad as it is, but things took a turn for the worse a week later when Google Mandiant researchers came out to say they found evidence Citrix Bleed had been exploited in the wild since late August, making the vulnerability one of this year's tens of actively exploited zero-days. Mandiant said a threat actor was using the bug to gain access to Citrix gateways and then pivot to internal systems. The attacks were small in scale and targeted professional services, technology, and government organizations.

"Things then went from bad to disastrous last week, around October 23-25, when several proof-of-concept exploits started popping up on GitHub and vulnerability portals. Within a day, mass exploitation was in full swing. At the time of writing, GreyNoise is tracking more than 120 IP addresses that are probing the Internet and attempting to exploit Citrix Bleed."

And finally, Kevin Beaumont later posted an update over on Mastodon. He said: "Quick update on Citrix Bleed. Tracking just over 20,000 exploited NetScaler servers so far today, where session tokens have been stolen. How? Have a honeypot running to gather data on attackers, then compare with Netflow via industry friends. Two TCP connections, first one large, plus Shodan cross reference to validate NetScaler victim. Also it turns out in March of this year somebody documented how to replay the session token to bypass multifactor authentication."

Okay. So on the back of last week's Cisco web portal disaster, we now have another web portal login authentication disaster from another manufacturer, Citrix, who has a similar long history of catastrophic security vulnerabilities. And Kevin has arranged to confirm that more than 20,000 instances of actual login session token exfiltration have occurred.

So now let's look at the mixed blessing arising from the need to patch a horrible and widespread problem in full public view. What steps do both curious security researchers and rapacious malicious criminals take to turn a patch into an exploit? The security research firm Assetnote explains. They wrote: "It's time for another round of Citrix Patch Diffing," as in differencing. "Earlier this month Citrix released a security bulletin which mentioned 'unauthenticated buffer-related vulnerabilities' and two CVEs. These issues affected Citrix NetScaler ADC and NetScaler Gateway.

"We were interested in CVE-2023-4966, which was described as 'sensitive information disclosure' and had a CVSS score of 9.4. The high score for an information disclosure vulnerability and the mention of 'buffer-related vulnerabilities' piqued our interest. Our goal was to understand the vulnerability and develop a check for our Attack Surface Management platform." So these guys wanted to reverse engineer it for their own security suite offering.

They wrote: "For those unfamiliar with Citrix NetScaler, it is a network device providing load balancing, firewall, and VPN services. NetScaler Gateway usually refers to the VPN and authentication components, whereas ADC refers to the load balancing and traffic

management features. We've covered issues in NetScaler several times before. We began by installing and configuring the two versions we wanted to compare. We chose 13.1-49.15 and 13.1-48.47. From our previous work with NetScaler, we knew to look in the /netscaler/nsppe binary. This is the NetScaler Packet Processing Engine, and it contains a full TCP/IP network stack as well as multiple HTTP services. If there's a vulnerability in NetScaler, this is where we look first.

"We decompiled the two versions of nsppe" - that would be NetScaler Packet Processing Engine - "with Ghidra and used the BinExport extension to create a BinDiff file." Okay. I'll pause here to mention that four years ago we discussed Ghidra at some length. It's a free and open source binary code reverse engineering tool which was initially developed by the U.S. National Security Agency, our NSA. The NSA initially released the Ghidra binaries during the 2019 RSA Conference and then followed up by releasing its source code a month later on GitHub. Before Ghidra, the proprietary and quite expensive IDA Pro was the only real game in town. Now that's no longer the case.

Okay. So resuming what Assetnote was explaining, they had said: "We decompiled the two versions of nsppe with Ghidra and used the BinExport extension to create a BinDiff file." You can probably guess, but "BinDiff" is short for "Binary Difference." And this, of course, is the key to the attack. When something is patched, some code somewhere is changed. So the first challenge is to locate the regions that the patch changed, then to carefully examine the before and after code to understand what problem that change was intended to fix.

They wrote: "This process takes a while as the binaries are quite large. To ensure success we tweaked the decompiler settings under Edit > Tool Options > Decompiler," and then they list the things that they changed. "After creating the BinDiff, we opened them up for comparison and found roughly 50, five zero, functions had changed. We proceeded to check each one, often opening both versions in Ghidra and comparing the decompiled output with a text diffing tool.

"We found two functions that stood out: ns_aaa_oauth_send_openid_config and ns_aaa_oauthrp_send_openid_config. Both functions perform a similar operation. They implement the OpenID Connect Discovery endpoint. The functions are both accessible unauthenticated via the..." and then they list two specific URLs where they respond respectively. Both functions also included the same patch, an additional bounds check before sending the response. This can be seen in the snippets below showing the before and after for one of those two.

Now, I have the two code snippets. I grabbed them and put them in the show notes because they're very clear, and anyone who codes will be able to see what's going on. But I can describe what's going on for those who are listening. I think everybody's going to get it. In the original code, the amount of data that is returned in response to the query - so basically what's happening is we have this OpenID query that's made over HTTP. So it's a standard HTTP style query that is asking for, that is addressing a specific URL known, when you're talking about web APIs as an endpoint, so it is making this query. It's got a bunch of, you know, standard query headers that you would expect with any HTTP query. And then it's going to get some response back.

So in the original code, the amount of data that's returned in response to the query is determined by the size of a string that results after substituting a whole bunch of variables into a string template. After all of those substitutions are made, the result will be a much larger string, so a 128K character buffer is provided to contain this expanded string after these substitutions are made. Basically there are small, you know, percent symbols in a template, and each of those percent symbols is expanded into a parameter from a parameter that is provided to this function.

The string substitution function is known as "snprintf," which will be quite familiar to C programmers. Snprintf performs the string substitution up to the limit of the length of the buffer that it's been given. Then it returns as its function return value the full length that the string would be after performing all of the substitutions. It will not overflow the size of the buffer it's been provided. But it will return a larger value than the size of the buffer it's been provided if the resulting substituted string will not fit into the provided buffer.

This is typically done to tell the programmer how large a buffer would be needed to hold the entire expanded string if the buffer that was initially provided was not large enough. Code that cared about that would then typically allocate a new buffer that was large enough, then perform the string substitution again into the now large enough buffer. But in any event, this is considered a safe string function that will not overflow the size of the buffer that has been provided.

So here comes the fatal flaw. In the original code, immediately after the snprintf does its work and returns the size of the required buffer, that size is directly used to specify the length of the data that will be returned from the user's query, returned to the user as a result of the user's or in this case an attacker's query. In other words, what the original coder who wrote this failed to do was to check that the size of the response was not larger than the buffer which had been set aside to contain it.

Leo: So you could easily put a 5R3, which is the size returned, is greater than hex 20,000, then truncate it or don't use it, but certainly don't substitute that number in.

Steve: Right.

Leo: Because you're going to have a buffer overflow.

Steve: And you will see exactly what you just said in the second snippet, Leo. Failing to perform that simple and required check meant that if the remote attacker could somehow arrange to cause that snprintf function to assemble a huge long string, and thus specify a massive response size, even though snprintf would not itself overflow the much smaller 128K buffer it had been provided, the response that would be sent back would be massive, with - and here it is - the data in that response coming from whatever happened to be lying around in RAM after that 128K buffer.

And, as it turns out, that just as with the infamous Heartbleed vulnerability, where the web server's private key just might have happened to be in nearby RAM, in the case of this Citrix Bleed disaster, the RAM following that 128K output buffer is chockfull of valid logged-on authentication tokens which the server dutifully sends back in response to the attacker's query, thus allowing any otherwise unauthenticated remote attacker to become authenticated by using those tokens and thus being logged on.

Leo: Wow. Easy mistake to make, though.

Steve: Yeah.

Leo: But you just have to get in the habit of checking these overflows. Wow. You could see the signing - you have to understand snprintf returns not the string, but the size of the string or what it would be if we hadn't truncated it.

Steve: Exactly.

Leo: And now they're reusing that size here.

Steve: Yes.

Leo: As the parameter for the VPN send response.

Steve: For the amount of data to send.

Leo: Yeah.

Steve: Exactly.

Leo: Mistake. And a lot of the time it would work.

Steve: Yup.

Leo: Till a bad guy gets a hold of it, yeah.

Steve: Yup. So, and that's one of the points we've often made here is that having code that works is entirely different from having code that is secure.

Leo: Sure, very important.

Steve: They are two completely different things.

Leo: And just as they said here in the fix, they check the size before they do that.

Steve: And they just skip the output. They just drop the response completely.

Leo: Wow.

Steve: So what Assetnote discovered in the patched and updated code, the change in the code that drew them to that spot was the addition of a simple check before sending any reply, to verify that the value returned by the preceding snprintf function was less

than 128K. In other words, the result of formatting the reply to the user would fit within the 128K buffer that was provided. And if not, the bogus query would be ignored, and no response would be returned to the user, or to any attacker.

And as for exploiting this, Assetnote explained, they said: "To exploit this, all we needed to do was figure out how to get the response to exceed the buffer size of 128K bytes. The application would then respond with the completely filled, meaning Citrix, would then respond with the completely filled buffer, plus whatever memory immediately followed the print_temp_rule buffer.

"Initially we thought the endpoint would probably not be exploitable. The only data that was inserted was the hostname, which is something that needed administrator access to configure. Luckily for us, we were wrong, and the value inserted into the payload did not come from the configured hostname." Get this. "It actually came from the HTTP host header." Okay, now, the host header, as we know, we talked about just recently, is one of the several query headers in any HTTP query. So it is entirely within the remote attacker's control. In other words, whoopsie.

These guys wrote: "We were also fortunate that NetScaler inserts the hostname into the payload six times, as this meant we could hit the buffer limit of 128K bytes without running into issues because either the host header or the whole request was too long. In other words, there was also a coincidental buffer size amplifier that worked to make attacks based on this mistake practical."

Leo: Wow. Wow.

Steve: So we might assume that the coder of the original implementation knew that the snprintf function was string safe, in that it could not be induced to overflow the buffer that was provided because the size of the buffer was also provided, and it would stop before it hit the end. But rather than using the actual size of the string that ended up being returned in the buffer, the coder made the mistake of using the returned value from the snprintf function, which is, as we now all know, not the same as the length of the string that it placed in the buffer. If the string that it wanted to place into the limited-size buffer was larger than the buffer, it would return that size. And unfortunately, the programmer used that value as the length of the reply to send back.

Echoing these thoughts, Assetnote concluded: "Here we saw an interesting example of a vulnerability caused by not fully understanding snprintf. Even though snprintf is recommended as the secure version of sprintf, it is still important to be careful. A buffer overflow was avoided by using snprintf, but the subsequent buffer overread was still an issue."

Leo: Yeah.

Steve: "Like previous issues with Citrix NetScaler, the issue was made worse by a lack of other defense-in-depth techniques and mitigations, not clearing sensitive data from what appear to be temporary buffers, and stricter validation on client-provided data being the two most obvious mitigations which could have been applied to minimize the damage."

So as a consequence of that small oversight which was apparently first discovered back in August, but also enabled by, as Assetnote observed, a more general lack of architectural care in a security appliance, the network integrity of more than 20,000 large and high-end enterprise and government users around the world who have these

network appliances has now been compromised. And not just found vulnerable, but actually known to have been exploited. And what's really, you know, as I've said, anyone can make a mistake. But it's possible to do this better.

For example, when I was working on SQRL's code, I had a window always up on the screen which continually and in real time, it was updated in the background, showed me the current RAM contents of all of SQRL's sensitive data. I had deliberately grouped it all into a single region for easier monitoring and wiping, and anything sensitive was wiped from RAM the moment it was no longer needed, and I was always able to confirm that visually. And, you know...

Leo: It's a good habit, yeah.

Steve: That's part of just doing it right. I was terrified that I would make a mistake. And people coding secure applications should live in a constant state of terror that they are going to make a mistake.

Leo: It's one of the reasons I like Lisp is because you can see the value as the program's running of all the variables at any given time. We don't have direct access to memory as you do in assembly, but that's a very useful tool. You know, as I think about it, this might have been a kind of an inexperienced coder that made the mistake because, if you think about it, the fact that snprintf returns the size of the data it received is a hint to the coder to pay attention. There's a reason why it returns that. It returns it so you can test it. I mean, it's explicitly there. And Lisp does that a lot, too, where if there's a side effect, you're going to get back a value that's going to help you, though, prevent bad effects.

Steve: One of the common coding patterns is to first call it with a buffer size of zero.

Leo: Yeah.

Steve: In which case you use it to tell you the size of the buffer it needs.

Leo: Oh, clever, yeah.

Steve: Then you allocate that buffer and call it again with that buffer, and you're [crosstalk].

Leo: That's probably what snprintf was designed to do.

Steve: That's exactly what it was designed to do.

Leo: Call it first. Don't do anything with the string. Get the size of it and then allocate that buffer. Oh, yeah.

Steve: Yeah, and in fact I was looking at this, thinking, well, this is one of the reasons everything has gotten so big and bloated. They've got a 128K buffer sitting around.

Leo: For no reason.

Steve: You know, just for, like, to have.

Leo: It's the case.

Steve: It's like, how lazy is that? Allocate it on the fly the size you need.

Leo: That's a good point. So maybe they didn't want to call snprintf twice. You know? But that's the whole point of it is it tells you what it's going to need.

Steve: Yes. You're right. You're right. You know, Leo, it's much better to have the security of 20,000 of your customers sacrificed.

Leo: Than run a function twice. Whoa. No, that makes a lot of sense. You return that value so people can use it.

Steve: Yup.

Leo: When a function returns something, you know, think about why it's returning that. That's good. That's really good. I like that. So a normal way to use snprintf would be call a test.

Steve: Call the null buffer first.

Leo: Yeah.

Steve: Yup.

Leo: Figure out what you're going to need.

Steve: It tells you how big a buffer it needs.

Leo: And call it again with the appropriate buffer size. And I would still test and say, what did you get back? Was it the same? Because if it wasn't I'd still want to, you know, make sure it wasn't overflowing.

Steve: Well, you are wearing suspenders, Leo, so...

Leo: And a belt, and the belt.

Steve: And I assume you have a belt.

Leo: Great stuff. You know, this is my favorite thing that you do, which is look at the code and see where the error happened. I know it's not always as easy as that. But I think that's a great, very instructional example.

Steve: It's a perfect learning opportunity, yeah.

Leo: Yeah, yeah, really great. Thank you, Steve. Have a great week. We'll see you next time on Security Now!.

Steve: Thanks, buddy. Bye.