SAD DNS

Description: This week the Chrome zero-days just keep on coming, and we contemplate what it means for the future. We have two interesting bits of ransomware meta news including a new tactic. We update after last week's Super Tuesday patch marathon, and examine new research into the most common source of Android malware to see where most unwanted apps come from and it's not what we would likely guess. We'll share a bit of listener feedback and an update on my work on SpinRite. Then we look at the new "SAD DNS" attack which successfully regresses us 12 years in DNS cache poisoning and spoofing attack prevention.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-793.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-793-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here with more Chrome zero-days, two of them in fact. Did you see the Facebook ad for Ragnar Locker, the ransomware? Steve will talk about that. We also have a story, a very sad story about an attack on DNS servers that's probably coming soon to a server near you. How to patch it and how it works, coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 793, recorded Tuesday, November 17th, 2020: SAD DNS.

It's time for Security Now!, the show where we cover the latest security and privacy news from the king of security, Mr. Steve Gibson. Actually, if you were the king you would be in trouble because you're doing a terrible job.

Steve Gibson: I could be the esquire.

Leo: The esquire.

Steve: That's right.

Leo: The guy who's protecting us is what he is. Hi, Steve. It's good to see you.

Steve: Yes, indeed. We have Episode 793 this week, and counting. Just crossed the middle of November. This one is titled "SAD DNS."

Leo: Awww.

Steve: And normally I'm tempted to talk about the acronym, like how that was made, but I don't even remember. It was such a stretch. They used, like, S for I don't know, something, spoof or secure or something. And then the A came from the beginning of a word, and the D from the last character of it. And I just, every time I saw it, I thought, well, okay, that is SAD indeed. But it's a compelling story. What we have is a - we haven't had one for 12 years, a functional DNS cache poisoning DNS spoofing attack. This, of course, harkens back to 2008 and Dan Kaminsky and his discovery. But we'll get to all that. Really interesting topic. And we're going to spend some time because what the guys did in order to pull this off is very cool.

First, we've got a continuing count of Chrome zero-days. We're going to contemplate what that means also for the future. We've got two interesting bits of what I would call "meta news" regarding ransomware, including a new tactic which has just come to light. We're going to update after last week's Super Tuesday patch marathon and examine new research into the most common source of Android malware to see where most unwanted apps come from. And it's not what we would likely guess. We're also going to share a little bit of listener feedback. I'm going to update on my work with SpinRite. And then we're going to plow into this really, well, it's well named, an awkward acronym, SAD DNS attack technology. And of course we do have a great Picture of the Week.

Leo: As always.

Steve: I think another good podcast for our listeners, indeed.

Leo: Let's get to our Picture of the Week. What do you say?

Steve: So we've got a six-frame cartoon here, two guys sitting side by side on a bench. The first one says, "Just think. At some point in my childhood, when my dad picked me up, it was the last time he would ever pick me up." The other guys goes, "Mmm." And the first guy says, "And neither of us knew it at the time. It's a sad thought, isn't it?"

Leo: It is. That's a sad thought.

Steve: And the second guy says, "Yeah." And then second guy says, "I sometimes think one day there will be some old programmer who will write the last line of PHP. And that'll be the end of years and years of PHP. And of course, he won't know it at the time, either. That's a sad thought, too; isn't it?" And then they're sitting in the fifth frame saying nothing. And finally the first guy says, "No." And the other guy says, "No, you're right."

Leo: Not sad at all.

Steve: Not at all sad. Ah, PHP, how we love to hate thee.

Leo: Oh, how funny.

Steve: Yes.

Leo: It's probably a thousand years off, though, I've got to tell you.

Steve: Yeah. Well, and really, think about it, you know, some hobbyists sometimes are writing a little bit of Pascal.

Leo: Oh, god, yeah.

Steve: I still actually have a career writing assembly language. So some things just really never die.

Leo: Every December, I don't know if you know about...

Steve: Leo, you and LISP.

Leo: I write LISP. That's as old as me. That's literally a language that's older than me. It was created in 1956. I do this every year. It's fun. I never get very far, but there's a programming contest called the Advent of Code. It's an Advent calendar. There's a new problem for every day in December. And every year, those of us who are enthusiasts, we gear up, we get ready, what language are we going to use. It's a great way to learn a language. They're very challenging problems. I saw somebody post yesterday, "I think I'm going to do it in Fortran." And I thought, oh, good luck. Have fun, my friend.

Steve: Well, and with Fortran you just get a piece of paper.

Leo: You can write it.

Steve: Get a piece of Scotch tape and put it down over the Shift key because...

Leo: You're going to be pressing it a lot.

Steve: ...Fortran didn't have any case.

Leo: All in caps.

Steve: It was all capitals.

Leo: Yup, all in caps, yup.

Steve: Back in the days of card punches. Boy, you know, memory was always a challenge. We were punching holes in paper in order to remember binary bits back then. So two more new zero-days revealed in Chrome. Last week we had three zero-days patched in the previous two weeks. Today we have five zero-days. Patched in the previous three weeks.

Leo: Geez.

Steve: I know. And we were just talking about this last week, saying, you know, once upon a time IE was the favored target. Now it's clear Chrome has become the majority browser. And it's trying to be kind of an everyman's application execution environment. It's trying to be a little mini operating system with all the crap that the World Wide Web Consortium keeps pouring into our browsers. And it has bugs.

So last Wednesday the 11th, Chrome announced the stable channel update for Windows, Mac, and Linux. We're now at 86.0.4240.198. And I had commented last week that I was already a dot one nine whatever it was, or 163 or something. I was further along than they'd made any announcement of, and I didn't know why. Maybe this was part of that. So this one is already rolled out. Under "Security Fixes and Rewards" in their announcement of this stable update, they noted with their standard boilerplate that details would be kept restricted until the majority of users would no longer be affected. They indicated that both of those new-in-the-wild zero days were discovered and reported by "Anonymous," the first on the 7th and the second on the 9th. And this thing was released on the 11th. So the update was pushed out to our desktops very quickly after it was reported to Google. And the bounty rewards for both of those was $TBD. So, you know, To Be Determined.

The first flaw was another of those "inappropriate implementation in V8," which is exactly the language that was used to describe the previous week's zero-day vulnerability. The other flaw was a user-after-free flaw in the site isolation component, which of course we depend upon because we don't want cross-site exploitability. And you know, this is the model for the way we need to be doing security moving forward. Researchers spot problems, either doing static research or by catching something that they see happening in the wild. They report them privately to the responsible party, whomever that is. That responsible party rewards them for their discovery and for keeping their report private, and then quickly updates the affected software, pushing it out to all affected parties or devices, depending upon what it is.

I mean, that's what we're seeing here. Problems are being found. I mean, they're going to exist in something as crazy complicated as a modern browser, not to mention an operating system. There's going to be problems. There seems to be no end of them. We'll be talking about last Tuesday's 112 things that were fixed. And remember, those didn't just appear in the last month. Those have been lurking in Windows and all related applications for probably a long time. We know that some of them affect Windows 7, and those are not getting fixed anymore. So, what, that's 2008. That's 12 years ago. So we have this problem.

One thing we know today with absolute certainty is that cyberwar and cybercrime, either ad hoc or organized, are very real things. They exist, and they probably will now forever. As with encryption, which we were talking about last week, once that genie is out of the bottle, you can't put it back. Fifteen years ago, when this podcast was just launching, some cyberwar may have been underway by nation states. I think I may have mentioned before that in the early days I was once approached by a three-letter agency and asked whether I might be interested in developing networked cybertechnology for offensive

purposes. I declined since that doesn't really appeal to me. And I naively failed at the time to appreciate that my country might actually be under attack, then or in the future.

But we're living in a science fiction world these days. We're all walking around with masks on, and we've got, you know, cyberwar is a real thing. But I'm sure that plenty of other people did not decline when they were approached. So this suggests that all of this has been percolating for decades. Stuxnet showed us what was possible. Edward Snowden filled in many suspicions and revealed a range of sobering truths about what our U.S. intelligence agencies had been and were up to.

So across this 15-year lifespan of this podcast, we've watched it happen. Cyberwar and cybercrime have gone from "Really, like that happens?" to "Oh, my god, take us off the 'Net now." There are corporations pulling the plug all over the place when they realize they're under attack. So it's now front of mind for many in the IT industry; and CSO, Chief Security Officer, is now a C-suite position, which used to not even exist.

And of course we know that not all software is in the same position. The firmware controlling an electric toothbrush just needs to manage its rechargeable battery, run a timer, and control its motor. If it's not connected, there's no way anyone can get to it. But any software, and I would highlight in bold "any," any software that is connected, even if it's a coffee pot or a thermostat or, as we know, printers, is exposed to what is obviously an increasingly hostile world. Nothing connected to the Internet can afford the luxury of being naive to the very real possibility that it might be used as an entry point for cybercrime. And no one using connected devices can afford not to exercise some caution over their deployment within a network.

And as I noted above, the Chrome project has evolved the right model. The browser is out on the front lines. It needs to be updatable autonomously with a super short cycle. But since Google has not done it right everywhere, we know that, Android is a catastrophe. And we'll be talking about that a little bit later. Yes, they're working hard to fix their earlier mistakes, making more of the Android OS push updatable. But the vast majority of today's Android devices, as we know, will never be updated. So we can only hope for the sake of their owners that those devices die as soon as possible. Just completely. Just remove them from the world, to be replaced, hopefully, by Google's newer autonomously updatable operating systems. That's where we have to be.

And as for the rest of the consumer and enterprise industry, the same goes. The only responsible way to produce anything that's connected to the Internet is to provide a built-in means for allowing the party responsible for its maintenance some kind of autonomous means for pushing out critical updates. Maybe get the permission of its user. You need a means to do that, though. Our routers don't really have that. They operate just completely autonomously. We've talked about this before. They've been a source of a great deal of insecurity. There just needs to be a way for these things to take care of themselves.

So anyway, I just, you know, the idea that every time we meet here Google has a few more editions of Chrome demonstrates that they've got that fast-cycle approach. They were told of one problem on the 7th, another one on the 9th. There's a new version on the 11th that removes them both. They're not going to create a perfect solution. We know that ship has sailed for anything that is sufficiently complex. So the only thing you can do is put in place a response cycle that fields problems, reacts to them quickly, and just closes them down, you know, keeping people as safe as possible.

A group known as Intel 471 published a report. I've got two pieces of sort of what I called "ransomware meta news." As we have unfortunately been observing, the ransomware "business," and I put it in quotes in my notes, is booming. I don't like calling it a business, but that's what it is. But we've been looking at this like one case at a time,

one piece of ransomware at a time. And so it occurred to me that it's easy to lose sight of the forest when we look too closely at the individual trees. When we pull back and take in the entire scope of what's been evolving somewhat unseen, what we have is a bit startling. These guys report on that.

The question they answer, how many Ransomware as a Service - that's our acronym, RaaS, Ransomware as a Service - operations are there today? We've touched on the comings and goings of a few of them. But a comprehensive review finds that there are presently more than 25 separate RaaS, Ransomware as a Service, portals now in business, in the business of renting ransomware for use by other criminal gangs. This group, Intel 471, published their report yesterday titled "Ransomware-as-a-Service: The pandemic within a pandemic," you know, talking about the whole COVID-19 thing and like this has been happening this year.

Their report was not highly detailed, but it was illuminating. And as I said, I disliked legitimizing this by calling it a business, but this is the way it's developing. I mean, yeah, it's dark. It's mal. It's criminal. But it's an enterprise. And I guess it's not really surprising that what would be revealed, and this was in their report, is a hierarchy of players. A few at the top who are pulling down the lion's share of the extortion revenue, they used the word "profit," but it just doesn't feel right to call it profit. So it's extortion-based revenue. There's also a much larger base of wannabe players who are in the "business." They've got portals. They're hoping to make their mark and then get cut in for a piece of this action.

So in looking through this report, I did note that this podcast had not missed anything. The big players noted by Intel 471's research are DoppelPaymer, Egregor, Ragnar Locker, REvil, and Ryuk.

Leo: They need some better vulnerability names. I know where you can get them, too.

Steve: So those are the guys at the top of the pyramid making the lion's share of the money. The one thing this research does show is that there are those other 20 me-too players who are working hard to move from the also-ran to the pay-us-now-or-else category. They may not stay as second-tier players forever. So we'll keep an eye on the news and see how this goes.

The second piece of meta news about ransomware, you just had to shake your head. Ragnar Locker took out a Facebook ad.

Leo: Wow.

Steve: I know. I know.

Leo: It's also, by the way, a pretty good name. I do like Ragnar Locker, yeah.

Steve: Ragnar Locker, yup. Last week we mentioned their successful ransomware attack on the Italian distiller Campari, and their publication at the time of the contract between Wild Turkey, one of Campari's brands, and Matthew McConaughey. It appears that this gang is into shaming their victims as a means of inducing payment. And I suppose this is the logical next step.

Leo: Sure.

Steve: Eventually companies are going to become more able to recover from the original, simpler, local ransomware encryption attacks; right? I mean, like they're going to, like, with this in the air as much as it is, and we've talked about the CEOs are going to be looking at their CIOs saying, "We can survive one of these; right? Everything's backed up? If this happens to us, you'll get us back online quickly?" And the CIO says, yeah, you know, we got that all set up last month or the month before, whenever. So it's foreseeable that the effect of the original "We'll give you the decryption key" extortion may be weakening over time.

So get this. After obtaining 2TB of sensitive data, which was stolen during their November 3rd attack, which was exactly two weeks ago today, and demanding a $50 million ransom paid in bitcoin, the Ragnar Locker gang have used a compromised Facebook account to take out a public Facebook ad threatening to publicly release the sensitive Campari data - remember they got 2TB of it - unless the ransom is paid. So apparently this new tactic is intended to publicly shame and pressure the ransomware victim into paying. Even if they don't so much need the decryption keys, they don't want it known that 2TB of potentially sensitive data is loose. The compromised Facebook account belonged to a Chicago-based DJ by the name of Chris Hodson. Chris believed that all of his online accounts had been protected by two-factor authentication. But it turned out that he missed one, Facebook.

Leo: Oh, geez.

Steve: Brian Krebs, who reported on this, wrote that Chris said a review of his account showed the unauthorized campaign reached approximately 7,150 Facebook users and generated an impressive 770 clicks, over 10%, with a cost per click of $0.21.

Leo: Wow. That's good.

Steve: Yeah. Chris said Facebook billed him $35 for the first part of the campaign, but apparently detected the ads as fraudulent before his account could be billed another $159 for the campaign. So we have a new wrinkle added to the ransomware scenario: Ransomware gangs who get into a corporate network are going to first exfiltrate as much data as they can. Then they will deny its owners access by encrypting it in their wake because why not? Encryption is easy now.

Then they contact the victim and demand payment, not only for providing the master decryption key, but also to threaten the release of their victim's sensitive corporate data with the newly added wrinkle that they might widely and publicly employ an advertising pressure campaign to up the ante and further coerce payment. And of course, you know, given that these attackers have already shown their true colors, I mean, right, they're criminals, there's no reason to expect nor any means to enforce the permanent deletion of the sensitive stolen data. So it's a mess.

On the other hand, we're sort of where we were before with these gangs having it in their own best interest, much as they were actually giving decryption keys when ransoms were paid. They have to honor their commitment to delete the data, or at least never release it, because if that ends up not being honored, then people will have no additional incentive to pay the ransom. So it's a weird world we're in, but it's the one we've got.

Last Tuesday's Windows patching, Microsoft fixed, as I mentioned, 112 known vulnerabilities. Not that there aren't going to be another 112 next month. But this month we've got this 112. Seventeen of those 112 were rated critical; 93 were important; and two were just moderate. And in terms of remote code execution, looking at it from that perspective, 24 of those flaws of those total of 112 created pathways for attacker-supplied code to be executed in vulnerable systems using Excel, SharePoint, Exchange Server, Windows Network File System - we'll talk a little bit more about that in a minute - Windows GDI+ component, and Windows print spooler service. Oh, and also Microsoft Teams.

We did receive the expected and hoped-for patch for the zero-day privilege escalation vulnerability which exists in Windows Kernel Cryptography Driver. Remember, that was the one where, because the driver exposed an API to the userland for use by applications, that created a connection from user mode down into the kernel. We learned about it the week before Patch Tuesday, when we learned that one of those zero-days that Google closed was part of a chain of which this was another part. Microsoft felt, well, since the particular exploit vector was the browser, and Google took care of that, you know, we expected it would get fixed last week, and indeed it was. So that's been fixed.

But this thing also affects Windows 7 and the synchronized server, Windows Server 2008 R2. I went over to 0patch.com, wondering whether they would be offering a fix for Windows 7 and the Server 2008 R2 folks. There was something in late October, but nothing yet in November. They may get around to it. And as the number of problems that Microsoft is not fixing on Windows 7 - which remember is still the number two platform in use. Windows 10, yes, finally did replace 7 in the number one slot, but Windows 7 is now number two. And so there's a lot of them, and they're never going to go away. They will be running Windows 7 until their hardware dies, much like those phones, the smartphones running Android, the old Android that's not being updated either, until those smartphones die. That's just the way this industry is turning out to be taking shape.

So I'll be keeping an eye on 0patch because, as I was saying, the longer this goes on, the greater the benefit to being covered by this third-party patching solution becomes for people who are still wanting to run Windows 7. And there are lots of places where you could run it safely, if it's not being exposed to the Internet, or if you're keeping a state-of-the-art browser. And Microsoft is continuing to keep Windows Defender updated, even on those systems. So you've got a lot of security. Although, yes, known problems that bad stuff that did happen to get in could take advantage of. And the number of those known baddies is continuing now to increase for those things that we know affect Windows 10, which also affect Windows 7. On the other hand, Microsoft is also introducing new problems in Windows 10 that will never affect Windows 7.

And in addition to repairing that zero-day and those many other repaired remote code execution vulnerabilities, an unspecified, and I love this, they just called it a "security bypass flaw," was fixed in Windows Hyper-V. But as I was writing that, it occurred to me that calling something a "security bypass" says absolutely nothing about it. Which I suppose was Microsoft's intention, since what cannot be described as a security bypass? Last week's Picture of the Week was that locked gate standing out all by itself in a field. That's a security bypass.

Leo: Yeah, you don't really need security to bypass that.

Steve: Yeah. One of the remote code executions is an RCE in the Network File System. It's rated 9.8 out of 10. And one place you really never want a remote code execution is in your network file system. Microsoft has also stated that the attack complexity is low,

which is only good news when you're the attacker. Since NFS runs over TCP and UDP port 2049, expect to be seeing an increase in port scanning targeting 2049. A Shodan query reported 38,893 exposed port 2049s on the Internet. However, exploitation requires that an NFS share be configured for anonymous write access, thus no authentication required. Doing something like that is not all that uncommon. Leo, I'm sure you'll remember how once upon a time FTP servers would often allow for anonymous file uploads into a specific protected directory.

Leo: Right.

Steve: You'd remove the privileges from the directory so that nothing there, like you couldn't execute anything from there.

Leo: Yeah, you could have a public FTP site, yeah.

Steve: Yeah, exactly. So it would collect stuff, and the idea being that such files received would be treated with caution, but that there was some valid reason for needing to receive unsolicited files.

Leo: Yeah, if you had a BBS

Steve: Like people submitting samples.

Leo: Yeah, or people uploading to your BBS, that kind of thing, yeah.

Steve: Yeah, exactly, exactly. So of those 38,893 servers answering on port 2049, we don't know what percentage of them are actually NFS shares and also configured for anonymous write access. But I'd put some money down on a bet that attackers are working to determine exactly what that count is right now. Microsoft also fixed critical memory corruption vulnerabilities in their scripting engine and in IE, and once again multiple remote code execution flaws in that HEVC, that's the Video Extensions Codecs library. As we know, codecs are interpreters. They're interpreting compressed media, and interpreters are hard to get right. And of course all of those have Internet-facing exposures. So yeah, as always, updating when you can would be wise.

Leo: Yeah.

Steve: I thought this was sort of interesting. Where do most malicious Android apps come from?

Leo: I'm going to guess the Play Store.

Steve: Oh, Leo. Yes. Good guess.

Leo: But that's because Android's not like a desktop operating system where you're downloading stuff all over the place. You kind of can, but almost everybody uses Android, that's where they put new stuff on their systems is the Play Store, yeah.

Steve: We have some numbers behind your correct intuition. It turns out the bulk of Android apps are not coming from unauthorized third-party app sources and repositories, as someone might think. In a recently published 17-page report titled "How Did That Get in My Phone? Unwanted App Distribution on Android Devices"...

Leo: I don't like it in my phone.

Steve: ...NortonLifeLock, which has now been - it's like the formal renaming of Symantec. Symantec is now NortonLifeLock. I guess they wanted just to go to products rather than...

Leo: Consumer products, yeah.

Steve: [NortonLifeLock] introduced the result of their careful four-month study. They wrote: "Android is the most popular operating system with billions of active devices. Unfortunately, its popularity and openness makes it attractive for unwanted apps, malware, and potentially unwanted programs (PUPs). In Android," they wrote, "app installations typically happen via the official and alternative markets, but also via other smaller and less understood alternative distribution vectors such as web downloads, pay-per-install services, backup restoration, bloatware, and Instant Messaging tools. This work performs a thorough investigation on unwanted app distribution by quantifying and comparing distribution through different vectors.

"At the core of our measurements are reputation logs of a large security vendor, which include 7.9 million apps observed in 12 million Android devices" - so big samples - "between June and September of 2019." So last summer, summer a year ago. "As a first step," they wrote, "we measure that between 10% and 24% of users' devices encounter at least one unwanted app, and compare the prevalence of malware and PUP," the potentially unwanted programs.

They said: "An analysis of the who-installs-who relationships between installers and their child apps reveals that" - as you said, Leo - "the Google Play market is the main app distribution vector, responsible for 87% of all installs and 67% of unwanted app installs, while also providing the best defense against unwanted apps. Alternative markets distribute instead 5.7% of all apps, but over 10% of unwanted apps." They said: "Bloatware is also a significant unwanted app distribution vector, with 6% of those installs. And backup restoration is an unintentional distribution vector that may even allow unwanted apps to survive users' phone replacement." In other words, you backed up a phone containing bad stuff, and then you restored it to a new phone and brought the bad stuff back with you.

They said: "We estimate unwanted app distribution via pay-per-install to be smaller than on Windows. Finally," they said, "we observe that web downloads are rare, but provide a riskier proposition even compared to alternative markets." And of course that's not a surprise. One of our longstanding rules of the road is, remember, never download something that is offered to you when you're surfing the web. Classically, oh, you need to update your version of Adobe Flash. Click here. Unh-unh, no, don't.

So anyway, exactly as you said, Leo, this research demonstrates that it's really just a numbers game. We've often said that installing from non-Google Play sources is dangerous. And that's true, as a percentage of per-installs. But the Google Play Store does a far better job of keeping the crap off people's phones on a percentage of the crap basis. When viewed overall, the Google Play Store being the source of 87% of all Android app installs simply means that, as their research showed, that's where two-thirds of all unwanted apps come from, only because that's where almost all apps come from. And of course, as a percentage of all apps, they're doing a better job.

But interestingly, this still supports and doesn't change our longstanding advice, which is don't install stuff just because it's there. That's where people get in trouble. Resist the temptation to say, "Oh, look at that," and then click, "Yeah, gimme." You know, some percentage will be junk that you don't want, even from the Google Play Store. And in fact, as the research shows, since there's so much more stuff there overall, the absolute amount of junk that is potentially unwanted on the Google Play Store is just that much higher because there's just so much overall, even though as a percentage it's lower. So anyway, that 17-page paper had tables and charts, and I have a complete decomposition of all this. I put a link to it in the show notes in case we have any Android users who are interested in looking at this in more depth.

I had two interesting bits of feedback that I wanted to share. Kevin Morris, he said: "Hi, Steve. I've been experimenting with various flavors of Linux, and once I burned an ISO to my flash drive, I always had a devil of a time ever being able to use it again." He said: "InitDisk solved the problem. Thank you for that. Kevin Morris." He's in Santa Clara.

Anyway, so I just wanted to use that as a reminder to me to remind our listeners that that little InitDisk utility that I created earlier in this phase of work for SpinRite, InitDisk will be the USB prep technology which I developed for this next version of SpinRite, since so much has changed since, was it 2004? Yeah, 2004, when SpinRite 6 was produced. And back then diskettes were still a thing. They're rare these days. Not gone, but rare.

Leo: They're pretty hard to find, yeah.

Steve: Anyway, yeah. InitDisk does such a good job of preparing a flash drive that, oddly, other things will refuse in some instances to work. InitDisk almost never refuses. And /dev/jake is his Twitter handle. He said: "@SGgrc, playing devil's advocate, but isn't the Let's Encrypt issue you spoke of last week a good reason to force users of Android devices older than 7.1 to upgrade, if they can, or abandon those insecure devices?" And of course he's talking about the problem of the Let's Encrypt root, which is expiring next year, and them scrambling a bit because they realize that fully one-third of Android devices which are older than 7.1 will never be updated.

So I just wanted to say, like, he says wouldn't it be a good reason to force users of Android devices older than 7.1 to upgrade? How do you do that? I mean, they own their devices. We can't, like, scare them, wave our hands around. They're obviously not paying attention to all this. They don't know what's going on. Stuff's just not going to work for them. So, I mean, I wanted to make sure that everybody understood that there's no leverage. Like wouldn't Microsoft love to force Windows 7 users to upgrade to Windows 10? Well, lord knows, Microsoft pulled every possible trick in the book, even removing the Close box from the dialog and only giving you a choice of now or later. They took away the "No, thanks" button finally, and still Windows 7 is the second most popular OS on the planet. Microsoft would like that number to be zero. They tried really hard.

So anyway, the point is that some things you can't make happen. This is one of them. It's just going to be those older devices. Their batteries finally just give up, they only

hold a charge for 10 minutes, and it's not a replaceable battery, so okay, fine, scrap heap for this thing. That's the only way they're going to go away, just like Windows 7 machines. You can't buy a Windows 7 machine now. Anything you get that has Windows on it is going to be 10. So over time, those old 7...

Leo: It's expensive, too. Not everybody can afford to buy a new phone like we do, or a new computer like we do.

Steve: Right, right.

Leo: Windows 7 you could update, but I understand why those old Android phones, a lot of them were low-cost phones to begin with, $50 phones to begin with. And you imagine a lot of them are in India and China and places where people just don't have the disposable income to replace them. Which is unfortunate, I mean, they're stuck with it.

Steve: What about hardware level and driver level? I would imagine a lot of them won't run as a very late-model Android.

Leo: Exactly, yeah.

Steve: Updating isn't even a...

Leo: Google created Android Go for these low-and-slow phones because they needed something that was lower resource use. It's a tough situation. But it's a testament to the success of Android, if nothing else. You don't have this problem with iPhones because there were no $50 iPhones.

Steve: Right.

Leo: Ever.

Steve: Right. And there aren't, I mean, as we know, Android is the largest install base.

Leo: Right, by far.

Steve: Billions and billions of those things.

Leo: Exactly.

Steve: So I thought it would be interesting for our listeners to share something that I posted yesterday in the spinrite.dev newsgroup at GRC. This was sort of - actually, it was just as I was putting things down to switch over to work on the podcast production. I

wrote: Everyone. Working with millQ - he's one of our very valuable contributors. That guy's got more PCs than anyone I've ever seen. Working with millQ through a series of tests, we finally tracked down the cause for drives appearing and disappearing, depending upon whether, even though the system was booted from diskette, an unused USB stick also in the machine was causing some hard drives not to be seen. A register within the PCI RAM space contained bogus data that my code had been relying upon. Since only four of that register's 32 bits should ever be non-zero, the addition of a sanity test to that register's contents now prevents it from being trusted when it should not be.

This resolved the drive coming and going that millQ was seeing, and it might also have fixed other known and as yet unknown problems from ever occurring. And I said: A good day for the future of SpinRite. I said: I'm switching to podcast prep mode until it's behind me. Thank you, everyone, for the patient testing and feedback. We're making some last improvements that are very useful. I said: millQ has some weird crashing behavior on that one very old laptop - he's got an old ThinkPad 390. And it turns out that by changing the setting of LASTDRIVE in the DOS config.sys he's able to cause some weird behavior. And we have a guy in Germany, Chris, has a hang problem on one AMD machine which also doesn't happen if he boots it from floppy, but does if he boots from USB.

Anyway, but, I mean, we're down to, like, almost no problems remaining. I was thinking, Leo, about the old - remember the old 90/10 rule where 90% of the work goes into the last 10%. I'm at the 99.99/0.01 rule.

Leo: Well, the 90/10 rule applies to every 10%. So at 99, the last percent is going to take 90 percent of the effort.

Steve: Ah, right.

Leo: You see? It's basically...

Steve: Exponential.

Leo: Yeah, you'll never get there, in other words.

Steve: Never really get it.

Leo: You'll never be done.

Steve: The good news is we're getting to the point where it's like, okay, folks. And a lot of the testers are really kind of getting annoyed because it has always worked for them on everything they've ever tried it on. But of course those aren't the people that I'm concerned about.

Leo: Right, right.

Steve: It's like, okay, good, yeah. But I want this thing, you know, we've got a lot of SpinRite 6 users in the world. And so every last little exponential 10% that I can remove from having a problem, the better. On the other hand, it's like, works for everybody.

Oh, and somebody else wrote to me, it was just a cute note. He said: "Dear Steve. I'm a regular follower of Security Now! and a VERY [all caps is his] happy SpinRite 6 user. It is my go-to tool to test new drives as I purchase them. I find that IF they successfully pass" - and this is a capital IF also, that's interesting, brand new drive - "successfully pass SpinRite's Level 4 testing, they usually give many good years of service.

"I'm writing to you today to ask you a favor. With your blessing, I would like to join your testing group for the latest SpinRite. Since it's nearly ready to go, what could one more tester hurt? I just purchased" - yeah. "I just purchased one of what will be four 4TB WD drives and found to my dismay that SpinRite" - and he's using 6. Oh, yeah, he said SpinRite 6 at the top - "that SpinRite doesn't do GPT" - that's of course the newer partition format, GUID Partition Table, he said - "in its current version." Duh. He said: "I suppose I should have known this, but didn't. If you could see fit to allow me into the program and use the current SpinRite beta, I promise not to blame you if I should ever develop an allergy to asparagus."

Leo: Okay.

Steve: Uh-huh. "For verification purposes, below is my original SpinRite purchase information. The original email address is no longer preferred as the company is defunct." He says: "Who could compete with TiVo? But you can reach me here on Gmail. With anticipation, thanks." So a couple things I need to clarify. The work we're doing is on the technology, like most of the technology, actually all of the new technology that SpinRite 6 will be needing to go to 6.1. And it's working. I mean, it's like, done, with one weird thing in Germany and a ThinkPad 390 or something, if you stand it vertically and shake it with a USB drive. I mean, it's like, okay, but I would like to know why. So everybody is welcome to work with us in the GRC newsgroups.

On the other hand, because they're kind of NNTP old-school text-only funky, that's why I created the GRC forums at forums.grc.com. This technology, and it's already calling itself ReadSpeed, that's going to be this benchmark, which is like the testing platform for the technology, it will be over on the web forums, which are much easier to use than the old-school textual NNTP newsgroup. But that's where all the development is always being done.

So if you're someone who, like, enjoys that kind of thing, you're welcome to join us over in GRC newsgroups. You just need to point a newsreader. You have to have a newsreader. Thunderbird can read newsgroups. Gravity is my favorite newsreader I've spoken of before. News.grc.com is the server. And welcome. But it's more friendly, user-friendly, and graphical. And the reason I created the web forums is because I wanted something that would be a better way for us to deal with everybody on the podcast who wants to play with this and then easily provide feedback. So that'll be there. And then that will evolve into the SpinRite web forums as we get there.

So what'll happen is ReadSpeed will get the last dents polished out of it, and we'll probably discover some when we get a much larger testing audience, which is my goal for this. Once that's done, then I switch into incorporating all of its technology into SpinRite 6 to create 6.1. And at that point we will have SpinRite 6.1 betas, and anybody who's a 6 owner will be able to use that purchase information that this guy was talking about, the serial number from your copy or your transaction ID from your purchase. And

if you have neither of them, you can write to Sue, and she'll look you up and get them. Those you then use to download the betas before we're officially finished.

So that's the whole deal in a nutshell, just as sort of a review to people. Basically, soon, I will be announcing on this podcast that this thing I've been working on is available for download by anyone over in the forums.grc.com. Come over, grab a copy. You don't need to join the forums to do that. You do need to join the forums in order to post and participate. But hopefully within a couple weeks.

Leo: Of course you can probably use SQRL to join those forums without any difficulty.

Steve: You can, as a matter of fact. And people have said the most painless joining experience they have ever had.

Leo: Back we go to the show. And I've got to show these people the logo for SAD DNS.

Steve: Awww.

Leo: Awww.

Steve: It's sad.

Leo: He's crying. He's sad.

Steve: So a team of researchers, as I said, from the University of California at Riverside and the Tsinghua University in Beijing, presented their paper at the ACM Conference on Computer and Communications Security, that's CCS '20, which was held last week. And it won the Distinguished Paper Award.

So, okay. Before we go any further, let's step back and review DNS cache poisoning, which this podcast covered in great detail when it first happened 12 years ago. That would have been, what, three years into the podcast. It was a biggie for us at the time. When a DNS resolver needs to look up the IP address of a domain on behalf of its user who has queried it, it in turn issues a query using the lightest weight means possible, which is a single UDP packet containing that query. The UDP packet is addressed in turn to a more authoritative DNS server, asking it about some component of the DNS domain name path. The reply packet, which is returned from the more authoritative DNS server, must be a reply returned to the same port on the first DNS server from which it was sent. And the reply's internal DNS transaction ID must match a query that the DNS server currently has outstanding.

So that all just makes sense. It was designed from the beginning. It's been there from the start. It was minimal. It was inherently trusting, as was all of the Internet back in those quaint old days. The whole system is admirably lean and very efficient. But it inherently suffers from a series of now quite well known vulnerabilities. For one thing, UDP is trivial to spoof. Unlike TCP, where the famous three-way handshake among other things verifies the IPs of each endpoint in a conversation, a UDP packet is simply

launched from the source IP it claims to have to whatever destination IP it chooses. And that packet can contain whatever it wishes. You know, just sort of like an Internet bullet that you can point wherever you want, and you're able to obscure where it came from.

So 12 years ago, during the summer of 2008 Black Hat conference, security researcher Dan Kaminsky presented his discovery of a massively widespread and critical DNS vulnerability that would have allowed attackers to send users to malicious websites, hijack email, and get up to all sorts of mischief. It allowed for the poisoning of a DNS resolver's cache, basically causing the cache to contain a wrong IP address for a given domain. Of course that would allow attackers to impersonate any legitimate website and steal data.

Now, okay, that was, what, 2008, so 12 years ago. A lot's changed since then. In today's world of HTTPS with certificate protection, this becomes more difficult. But if a DNS cache is poisoned anywhere upstream of where Let's Encrypt, for example, does its lookups, then it's trivial to again obtain certificates for any desired website. The flaw that Dan found allowed for arbitrary DNS cache poisoning, which at the time affected nearly every DNS server on the planet. I say "nearly every" because the one brand of DNS - one - that was not vulnerable had been created by our old friend Daniel J. Bernstein, who had nine years earlier, back in 1999, told everyone not to take the mistake that they did, but no one paid attention. He wrote his own DNS server, he did it right, and it was not vulnerable to this attack that Dan had found.

When Dan realized, Dan Kaminsky realized the extent of the vulnerabilities - that is, virtually all DNS servers could be easily poisoned - and what that meant for the integrity of the Internet, which depended, as it still does today, upon the integrity of DNS, he called Paul Vixie. Paul was the well-known creator of several DNS protocol extensions and applications used throughout the Internet. He told Paul about the bug that he'd found. Together, they then called an emergency summit to Microsoft headquarters to discuss how to address this widespread problem and pulled in all of Paul's many contacts and vendor representatives. That coalition worked silently to create a fix for the vulnerability before it was disclosed to the public. All of the vendors simultaneously released patches for their products on July 8th of 2008. I mean, this was that big a deal.

However, according to Dan at the time, they didn't actually repair DNS. They just took a 16-bit transaction identifier which, for reasons of the protocol, could not be expanded beyond 16 bits. And they added random UDP source ports as a hack to expand the query entropy from 16 bits to 32 bits. In other words, DNS queries used to be issued from the fixed DNS service port 52, and the query contained a 16-bit transaction ID whose reply needed to match. Even if the transaction ID was randomized, that's still only 64K possible transaction IDs. So an attacker could stuff 64K replies into an awaiting DNS server, and one of them would be accepted as valid when it matched the transaction ID that was outstanding, thus poisoning the server's cache for anyone who then asked for that reply's DNS IP.

But by simply randomizing the DNS query's source port, instead of always issuing it from port 52, the change the entire industry made all at once on one day, by switching it to a random query source port, now any successful reply, that is, a successful attacker's reply, would need to match both the port and the transaction ID, thus expanding the query entropy from 16 bits to 32. Once that had been done, an attack that used to take 10 seconds, that's what Dan had, and he demonstrated it to the group privately to get their attention, took 10 seconds to spoof a DNS query, going from 16 bits, which is, as we know 64K possibilities, to 32, which is now 4.3 billion possibilities, well, that no longer made random guessing attacks possible. It eliminated this instantaneously successful attack. And the industry breathed a collective sigh of relief about the bullet that they had dodged.

Okay. As we know, just wanting to have systems upgraded doesn't make it so. Therefore, coincident with that and the podcast when we talked about it, and also in the spirit of the ShieldsUP! service, I created GRC's DNS Spoofability Service. It allows users to quickly and easily check the DNS servers they are using by virtue of their own Internet connection, typically their ISPs, but more recently any of the other popular third-party providers. It analyzes the DNS query transaction entropy of those servers. And back then, shortly after this had all happened, many users were still discovering that their DNS provider was using vulnerable DNS resolvers that were generating easily spoofable DNS queries.

The good news is, that is, that was 12 years ago. That's much less so today. It's rare that you run across a very poorly written server. Okay. So that was 12 years ago, 2008. Things have been relatively quiet since that flurry of frenzied activity to dodge that bullet. Until now. This new group of researchers have titled their award-winning paper "DNS Cache Poisoning Attack Reloaded: Revolutions with Side Channels." Their paper explains their accomplishment with its introduction.

They wrote: "In this paper, we report a series of flaws in the software stack that leads to a strong revival of DNS cache poisoning," they said, "a classic attack which is mitigated in practice with simple and effective randomization-based defenses such as randomized source port. To successfully poison a DNS cache on a typical server, an off-path adversary" - "off-path" meaning an adversary that's not participating in the local network traffic - "would need to send an impractical number of" - and they say 2^32. Actually on average half that, right, but still, that scale - "2^32 spoofed responses, simultaneously guessing the correct source port," which is 16 bits, "and the correct transaction ID," also 16 bits.

They said: "Surprisingly, we discover weaknesses that allow an adversary to 'divide and conquer' that space by guessing the source port first, followed by the transaction ID. This leads to only 2^16 plus 2^16 spoofed responses." Otherwise it would have been 2^16 times 2^16. You'd much rather have a plus there than a times. They said: "Even worse, we demonstrate a number of ways an adversary can extend the attack window" - which is to say that the time available to make all these guesses - "which drastically improves the odds of success." And I had in here in my show notes a reminder of a time that we saw this before. Remember that horrible vulnerability that was discovered in the - I wrote in my notes "WPA protocol," but it was actually - it was an aspect of it. It was where you had the eight-digit PIN. So was that WPS?

Leo: That was WPS, yeah, where you press a button to - yeah, I remember that, yeah, yeah.

Steve: Right. So you had an eight-digit PIN. Now, normally that would be 100 million combinations and be impractical to guess. But it was discovered that the first four digits could be guessed first independently and verified.

Leo: So dopey.

Steve: Bringing that down to 10,000. And it turns out, since the last digit of the eight was a check digit, then the final three, which is all you really had because you could calculate the last one, they could then be guessed and verified. So you'd have a thousand. So the guessing space was reduced from 100 million to 11,000. The first 10,000 plus the second 1,000, so 11,000. In other words, divide and conquer. So we've seen this kind of thing before.

They wrote: "The attack affects all layers of caches within the DNS infrastructure, such as DNS forwarders and resolver caches, and a wide range of DNS software stacks, including the most popular BIND, Unbound, and dnsmasq, running on top of Linux, and potentially other operating systems. The major condition for a victim being vulnerable is that an OS and its network is configured to allow ICMP error replies." And I'll explain exactly because this is very - this is diabolically clever.

They said: "From our measurement, we find over 34% of the open resolver population on the Internet are vulnerable, and in particular 85% of the popular DNS services, including Google's 8.8.8.8" and also, elsewhere, 1.1.1.1, both vulnerable. "Furthermore, we comprehensively validate the proposed attack with positive results against a variety of server configurations and network conditions that can affect the success of the attack, in both controlled experiments and with a production DNS resolver." And they take pains in their paper to say "We got permission to do that. You know, we didn't do this without permission."

So their attack disclosure page provides a link to test one's own DNS servers. So while I was preparing this report, I clicked the link, and I received "Your DNS server IP is 172.68.191.21. It seems your DNS server is running Linux later than 3.18. Since it is running the vulnerable version of OS that has not been patched yet, your DNS server is vulnerable. The test currently only takes the side channel port scanning vulnerability into consideration. A successful attack may also require other features of the server, for example, a supporting cache." Then it says: "This test is conducted on 2020-11-16" and the UTC time and date and so forth.

So what does this mean? Separate from their report, their attack's description web page lays it out. They said: "'SAD DNS' is a revival of the classic DNS cache poisoning attack," they said, "which no longer works since 2008, leveraging novel network side channels that exist in all modern operating systems including Linux, Windows, macOS, and FreeBSD. This represents an important milestone, the first weaponizable network side channel attack that has serious security impacts. The attack allows an off-path attacker to inject a malicious DNS record into a DNS cache, for example, BIND, Unbound, dnsmasq. The SAD DNS attack allows an attacker to redirect any traffic originally destined to a specific domain to his own server and then become a man-in-the-middle attacker, allowing eavesdropping and tampering of the communication."

Okay. So that's sort of the broad strokes. A bit deeper into their paper they lay out in some greater detail what they've done. And this is what is so cool. So as I said, once upon a time the port from which a query was generated was fixed at DNS port 52. Was it 52 or 53? Now I'm doubting myself. But anyway, one of those two. So the only random component of the query was the transaction ID. That's what Dan realized, realized it was too small. And so they started randomizing the source port of the DNS queries to get a 32-bit entropy for the query.

What these guys have figured out is a way to use ICMP as side channel feedback to derandomize the query port, that is, figure out the query port. That collapses the guest space back to 16 bits, like it was in 2008. And we now have, once again, practical DNS cache poisoning. So in the depths of their paper they said: "Historically, the very first widely publicized DNS cache poisoning attack was discovered by Kaminsky in 2008, who demonstrated that an off-path attacker can inject spoofed DNS responses and have them cached by DNS resolvers. This has led to a number of DNS defenses being deployed widely, including source port randomization and other defenses such as DNSSEC. Unfortunately, due to reasons such as incentives and compatibility, these defenses are still far from being widely deployed." He didn't mean randomization. Everybody does that. He meant like DNSSEC. Or they meant DNSSEC.

"To summarize, source port randomization becomes the most important hurdle to overcome in launching a successful DNS cache poisoning attack. Indeed, in the past there have been prior attacks that attempt to derandomize the source port of DNS requests. As of now, they are only considered nice conceptual attacks, but not very practical. Specifically, one requires an attacker to bombard the source port and overload the socket receive buffer, which is not only slow and impractical, unlikely to succeed in time, but can also be achieved only in a local environment with stringent RTT [round trip time] requirements. It is assumed that a resolver sits behind a NAT which allows its external source port to be derandomized, but such a scenario is not applicable to resolvers that own public IPs."

They said: "In contrast, the vulnerabilities we find" - that is, that they have found and demonstrated - "are both much more serious and generally applicable to a wide range of scenarios and conditions. Specifically, we're able to launch attacks against all layers of caches which are prevalent in the modern DNS infrastructure, including application-layer DNS caches, for example, in browsers; OS-wide caches; DNS forwarder caches, for example, in home routers; and the most widely targeted DNS resolver caches.

"The vulnerabilities also affect virtually all popular DNS software stacks, including BIND, Unbound, and dnsmasq, running on top of Linux and potentially other OSes, with the major requirement being the victim OS is allowed to generate outgoing ICMP error messages." They said: "Interestingly, these vulnerabilities result from either design flaws in UDP standards or subtle implementation details that lead to side channels based on a global rate limit of ICMP error messages, allowing derandomization of source port with great certainty."

They said: "To demonstrate the impact, we devise attack methods targeting two main scenarios, including DNS forwarders running on home routers, and DNS resolvers running BIND and Unbound. With permission, we also tested the attack against a production DNS resolver that serves 70 million user queries per day, overcoming several practical challenges such as noise, having to wait for cache timeouts, multiple backend server IPs behind the resolver frontend, and multiple authoritative name servers. In our stress test experiment we also evaluate the attack in even more challenging network conditions and report positive results.

"In this paper, we make the following contributions: We systematically analyze the interaction between application and OS-level behaviors, leading to the discovery of general UDP source port derandomization strategies, the key one being a side channel vulnerability introduced by a global rate limit of outgoing ICMP error messages. Two, we research the applicability of the source port derandomization strategies against a variety of attack models. In addition, to allow sufficient time to conduct the derandomization attack, we develop novel methods to extend the attack window significantly, one of them again leveraging the rate limiting feature, this time in the application layer. And, finally, three: We conduct extensive evaluation against a wide variety of server software, configuration, and network conditions, and report positive results. We show that in most settings, an attacker needs only minutes to succeed in an end-to-end poisoning attack. We also discuss the most effective and simple mitigations."

Okay. So the crux of what they have done is to arrange to eliminate the entropy creating query source port randomization. As I said, if the DNS query source port can be determined, then the effectiveness of DNS spoofing and cache poisoning attacks return to their quite tractable pre-Kaminsky severity. The reason ICMP comes into the picture is that an incoming UDP packet hitting a closed port will evoke an ICMP "port unreachable" error response, whereas the same packet hitting an open port won't. Thus it's possible to probe a DNS server's ports by monitoring ICMP replies. But it's not that easy, since all modern operating system IP stacks have evolved to deliberately rate limit ICMP error replies.

Old-timers among us will remember those good old days where it was possible to ping a server, an echo request ICMP, with a spoofed source IP. And that server would dutifully reply with an ICMP echo reply to the apparent incoming source IP. That's where the first DDoS attacks came from. You would flood a whole bunch of machines out on the Internet with ping packets, ICMP ping packets having the IP of your target victim. Those packets would bounce off of those servers, which would send ICMP echo replies all to the victim and flood their connection bandwidth. Anyway, you can't do that anymore because there is per-IP rate limiting, and also whole server, in other words, global rate limiting on all now IP stacks.

Okay. So how did these guys get around it? This is what is just so clever. They wrote: "Even though a source port can be directly probed by any attacker IP, in this case for example as in unbound and dnsmasq," they said, "it is imperative to bypass the per-IP rate limit," they said, "present in Linux primarily, to achieve faster scan speed." They said: "We develop three different probing methods that can overcome the ICMP rate limit challenge."

They said: "First, if the attacker owns multiple IP addresses, either multiple bot machines or a single machine with an IPv6 address, then it is trivial to bypass the per-IP limit. IPv6 address allocation states that each LAN is given a /64 prefix, effectively allowing any network to use 2^64 public IP addresses. We have tested this from a machine in a residential network that supports IPv6 and picked several IPs within the /64 to send and receive traffic successfully." So that works.

Or: "Two, an attacker who owns only a single IPv4 address, it is still possible to ask for multiple addresses using DHCP," which I didn't know. They said: "We verified that multiple private IPv4 addresses can be obtained on a home network. In addition, we've tested this in an educational network, where a single physical machine is able to acquire multiple public IPv4 addresses through this method, as well."

And, finally: "Three, if an attacker owns a single IPv4 address, and the above methods fail for some reason, for example, statically assigned IPs, then the last method is to leverage IP spoofing to bypass the per-IP rate limit and the global rate limit as a side channel to infer whether the spoofed probes have hit the correct source port or not, in other words, with or without ICMP responses." And here's what they came up with. And this is just what happens when a bunch of smart people are faced with a problem that nobody has had before; and, looking at what they know and the way the system works, they then engineer or invent, choose your term, a solution.

They wrote: "As has been shown in the context of TCP recently, global rate limiting can introduce serious side channels. Here we leverage the ICMP global rate limit to facilitate UDP port scans, which we describe next." And it's a little creepy because this could generally be used for UDP port scanning on the Internet. They said: "In observing the maximum globally allowable burst of 50 ICMP packets in Linux, the attacker first sends 50 spoofed" - that is, we're talking about spoofed source IP - "50 spoofed UDP probe packets, each with a different source IP, thus bypassing the per-IP rate limit." Right? Because they're all going to be from a different IP. "If the victim server does not have any source port open among the 50, then 50 ICMP port unreachable messages will be triggered," and they'll be sent out.

And they say: "But they are not directly observable to the attacker since each of the 50 will be returned to one of the spoofed source IPs. If the victim server does have open ports, then only" - it says five here, but it must be 49. If it does have open ports, then "49 ICMP packets will be triggered, as the UDP probing packets will be silently discarded at the application layer." He says: "Now the attacker sends a verification packet using its real IP address, a UDP packet destined to a known closed port, such as 1. It will either

get no response, if the global rate limit has already been met and thus drained, or an ICMP reply otherwise."

Okay. "Thus," they write, "even without receiving replies to the first 50 probes, the presence of an open port within a range of those 50 can be directly determined by inference. If no port is found in the first batch, the attacker waits for at least 50 milliseconds for the rate limit counter to recuperate and reset, and then starts the next round," using the next sequential set of 50.

They said: "Effectively, the scanning speed will be capped at about 1,000 per second. It therefore takes a little more than 60 seconds to enumerate the entire port range of 65,536 ports." Actually, it's 65,535, of course. "Nevertheless," they said, "it is a winning battle as the attacker can simply repeat the experiment, and the probability that one experiment will succeed increases dramatically."

Okay. So what these guys have done is to figure out a way to use modern ICMP rate limiting, essentially against the system that is rate limiting, as a side channel to probe a DNS server for its open port. So what this does is it moves the DNS spoofability clock back 12 years to a point where DNS cache poisoning sent the industry into a frenzy. Just incredibly cool research.

As I mentioned above, these days things are less panic-prone because the entire industry has switched over to HTTPS and to a certificate-based system to add authentication to privacy. And of course in the process, you know, it's difficult to get a certificate for a domain. You need to have a cert if you're going to spoof something that's on HTTPS. But as I also noted, automated certificate services are already existing, and more are coming online. So the practical vulnerability, while not hair on fire, is still there. We haven't yet secured our domain name system, and it doesn't appear that's going to happen anytime soon. So this is less urgent than it is supremely clever. But I wanted to share it with everyone just for its cleverness.

They have a Q&A on their page. They ask: "Am I affected by the vulnerability?" And answer: "Likely, as long as you are using a vulnerable DNS service." And then they suggest 8.8.8.8 and 1.1.1.1.

Leo: Although I just checked Cloudflare, and 1.1.1.1 has been mitigated. Cloudflare has jumped right on it, of course.

Steve: As we would expect them to. Very cool.

Leo: And they actually have an excellent blog post on this.

Steve: Oh, cool.

Leo: And I also checked NextDNS, which is the DNS server I use, which I really love, and they also are mitigated. So you should check with your provider. And if it's Comcast, god bless.

Steve: Yeah, and there is a link on that page that allows you to make a quick check.

Leo: Yeah, yeah. Sorry.

Steve: So they said: "How widespread?" According to their measurements, "35% of open resolvers are vulnerable to the attack, 'open resolvers' meaning a resolver which will accept queries from the Internet." They said: "We also found 12 out of 14 public resolvers and four out of six routers made by well-known brands to be vulnerable." In theory, any DNS server running the newer version of popular operating systems without blocking outgoing ICMPs, and they said only Windows blocks it by default, is also vulnerable.

So they asked: "Has SAD DNS been patched?" They said: "Yes, we have worked with the Linux kernel security team and developed a patch that randomizes the ICMP global rate limit" - which is a clever solution - "to introduce noise into the side channel. Please refer to Security Advisories for more recent updates." In other words, if you can't count on it being 50, if Linux now changes its mind from 35 to 100, for example, then you've lost your side channel. They've made the side channel much noisier. You can't count on that 50 being a hard number. Which makes it much more difficult to, well, arguably nearly impossible to perform that enumeration.

They said: "Which versions of operating systems are affected? Linux 3.18 through 5.10. Windows Server 2019, which is version 1809, and newer." They said: "We did not test older versions. macOS 10.15 and newer. We did not test older versions. FreeBSD 12.1.0 and newer. We did not test older versions." They said: "The patch for Linux is integrated into 5.10 and backported to many stable versions. However, we don't know how and when Windows, macOS, FreeBSD will patch this vulnerability."

Leo: Now, this has to be running on the DNS server, not on your - it's not your system that would be patched. It's the DNS server system; right?

Steve: Correct. It's the DNS server which would no longer be issuing its queries.

Leo: So it seems highly unlikely that many people are running Mac or even Windows DNS servers. I'm sure they're almost all Linux.

Steve: Yes. And this is more of an attack against public servers on the 'Net, which would then be fooling people who relied on those public servers, redirecting them to a bad guy's...

Leo: So they don't attack you. They attack the server, hoping that it will then be spoofed when you go there, and you can get a bad guy site instead of Google.com.

Steve: Correct. Anyway...

Leo: It's really interesting, yeah.

Steve: Yes.

Leo: And you would think, well, people must keep their Linux servers up to date. But no, if you're running BIND, or unbound on a Linux server, you probably don't update very often at all. That's why it's important to backport to stable...

Steve: Yeah, just leave it alone, you know. If it's not broke.

Leo: Yeah. 5.10 is very current. So interesting, yeah. As I said, great post at Cloudflare. They say they have mitigated it. And it says: "As part of a coordinated disclosure effort, earlier this year the researchers contacted Cloudflare and other major DNS providers."

Steve: Good.

Leo: So they have had plenty of time before this ACM conference to fix it. But I bet you that a lot of the big ISPs like Comcast - I wonder.

Steve: Not Cox. Cox is the one I tried last.

Leo: I don't think that test is completely definitive, though.

Steve: No, I think you're right. I think all it's doing is doing a version check and saying, okay, this is a known vulnerable version of the underlying OS.

Leo: Because they don't have time to do 50 pings from 50 IP addresses and all of that.

Steve: No.

Leo: They're not actually attacking. I would hope, anyway. Very good.

Steve: This is the way we keep our Internet locked down is this kind of research by clever people writing papers and, again, responsibly disclosing, letting these things get fixed.

Leo: Exactly.

Steve: And then they get to have the fun of saying, "Look what we figured out."

Leo: Exactly. Exactly.

Steve: Very cool.

Leo: Yeah. I'm glad you talked about this.

Steve: Now we have Happy DNS.

Leo: Yes. One hopes they're using Happy DNS from now on.

Steve: Turn that frown into a smile.

Leo: My friends, we've come to the end - I can't believe it, it goes so fast - of this particular episode of Security Now!. You will find Steve Gibson and the show at his website, GRC.com. Now, the first thing you should do when you get there is buy a copy of SpinRite, if you don't already have one. That way you'll be in on the upgrade to 6.1 the minute it's available. And as Steve mentioned, you could participate in the beta testing and the forums and all of that. But you've got to be a licensed holder of a copy of SpinRite, so you want to get that now. Free upgrade to 6.1 when that comes, for those who buy now.

You'll also find the show there, the 16Kb audio version for the bandwidth-impaired or those who don't really have very good hearing. If you have a little bit better hearing, you might want the 64Kb version. It sounds a lot better. He also has perfect fidelity in the transcripts written by Elaine Farris, 100% perfect fidelity. Those are great to have for two reasons. One, you can read along as you listen, which really helps, at least for me. I like to read the show notes as you're going through this. The transcripts are even better. But second, also, it is a great way to search. Because they're searchable, and they're plaintext, it's a good way to find anything in any of the 793 episodes of Security Now!. So take advantage of that, GRC.com, and all the other freebies that Steve has. Read up on Vitamin D, by the way, a lot of evidence now that Vitamin D is very helpful in fighting COVID-19. I just saw another study came out today. So read up on Vitamin D.

Also our site has copies of the show, 64Kb audio and video at TWiT.tv/sn for Security Now!. There's a YouTube channel devoted to Security Now!. You can watch or listen there. Best thing to do would be get a podcast application and subscribe. That way you get it automatically, the minute it's available. There are lots of those out there. Steve, have a great week.

If you want to come back and watch us do this next time, we do it right after MacBreak Weekly. This is a big long day, so we sometimes don't get to it till 1:30 or 2:00 p.m. Pacific. That'd be around 5:00 p.m. Eastern time, 21:00 UTC. Thank you, Steve. We'll see you next week. I can tell, he's giving me the salute. He's ready to go. We'll see you next time on Security Now!.

Steve: Bye, everybody. I'm ready to read my Peter Hamilton.

Leo: I thought you might be.

Steve: Woohoo.