Sunburst & Supernova

Description: This week, as we end 2020, we look at Chrome's backing away from a security initiative; Firefox's move to further thwart tracking; all of the browsers once again saying "No!" to Kazakhstan; the formation of a new industry-wide Ransomware Task Force; this week's widespread WordPress security disaster; the return of Treck's insecure embedded TCP/IP stack; and, yes, finally, the long-awaited announcement of the release of the ReadSpeed Benchmark which serves as a testbed and proof of operation for the next generation of SpinRite. Then we look at everything more that has come to light three weeks downstream from the first revelations of the SolarWinds-based massively widespread network intrusion and compromise.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-799.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-799-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson's here. Yes, we're doing one last show in 2020, and we've got to wrap up the biggest story of 2020, the biggest hack of living memory, which means probably the biggest hack of all time. He's going to give us the update on SolarWinds and Sunburst and Supernova, plus a lot of security news. Yes, that's one area where the news never sleeps. Steve Gibson is next.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 799, recorded Tuesday, December 29th, 2020: Sunburst & Supernova.

It's time for Security Now!, the last show of 2020. Yes, we have a brand new live show with this guy right here, Steve Gibson of GRC.com.

Steve Gibson: And I think everybody is unified in saying good riddance to 2020.

Leo: Oh, my god.

Steve: What a horrific year this has been.

Leo: I can't remember a worst year. Just awful.

Steve: It's too bad, too, because I like the number. It's all round and nice and clear sight, 2020. But oh, my goodness.

Leo: It should have been a good year. It should have been.

Steve: Yeah. So we're going to look this week at a bunch of stuff. We do have a lot of news because of last week's best-of. I went back all two weeks from now looking for everything that had happened since. And, not surprisingly, a bunch of stuff happened. Chrome is backing away from a security initiative of theirs for interesting reasons. Firefox is moving to further thwart tracking, using again a really interesting, cool technology. All of the browsers are collectively once again saying no to Kazakhstan. We have the formation of a new industry-wide Ransomware Task Force, which is just good to see happening because of the scourge that's been this year. We have, of course, the week's widespread WordPress security disaster du jour; the return of Treck's insecure embedded TCP/IP stack.

And, yes, finally, the long-awaited announcement of the release of the ReadSpeed Benchmark that I've been working on basically for six months, ever since I finished working on the USB boot prep stuff. So basically here we have this year's work product ready for release. And of course it serves as a testbed and proof of operation for the next generation of SpinRite. And then we're going to conclude by looking at everything more that has come to light three weeks downstream from the first revelations of the SolarWinds-based, massively widespread, network intrusion and compromise. So this Security Now! Episode 799, perfectly numbered for the end of the year, is "Sunburst & Supernova." And I think another great podcast for our listeners.

Leo: Can't wait.

Steve: We even have a, you were probably going to just say, also the bonus of it being a sponsor-free podcast.

Leo: Commercial-free. You know how at the end of the year all the radio stations do "And now another 99 minutes of rock and roll, commercial free, without interruption." Let's tell you the honest truth. Nobody's buying ads this time of year. So it's not because we love you. We do. But no, we're not going to pause, we're just going to roll right through with our Picture of the Week.

Steve: And what's interesting, Leo, is that I would imagine our listenership goes up this time of the year because people have more available time to listen to...

Leo: Don't get me started on the crazy wherefores and whys of advertising. It's a nutty business, nutty, nutty business.

Steve: As you said, we do have just a fun Picture of the Week. This, just based on the typography, anyone who's ever been a New Yorker reader will recognize that this really looks like it came from The New Yorker, although CartoonStock.com and CartoonCollections.com have their little watermarks on it. But anyway, it's got two kids running a lemonade stand. And normally the lemonade stand will have the price posted on it somewhere; right? It'll say "10 cents," or I guess more recently "25 cents." Anyway, this lemonade stand says nothing, just says "Lemonade." And then in the foreground we

have a man and a woman. He's holding one of the cups, and apparently he's telling her, "It's free, but they sell your information."

Leo: Oh, god. That's a perfect cartoon for 2020.

Steve: It is, indeed.

Leo: Now, I should tell you, because you'd normally need some breaks during the show for caffeination or hydration, that if you want I can do some jumping jacks, sing a song, do some tap dancing. At any point, I'll be here. You just say "Help me, help me," and I will take over.

Steve: And Leo, unless they're listening to it on the live stream, they probably have a pause button that they could press.

Leo: Oh, that. Oh, yeah.

Steve: If nature were to call.

Leo: Oh, good point, yeah. Never thought of that.

Steve: However, they will not be needing their fast-forward button in this episode, so we can get right in here. We've got Chrome 87, which it turns out is quickly backing away from insecure form warnings. We've had some fun from time to time at Google's expense with the way they tend to roll out new features that might have an impact on some of their visitors. The most recent of these was that very gradual deprecation of the Chrome browser's FTP services, remember, where first only 1% of Chrome users were going to be denied FTP URLs. It was like, really? Anybody still uses FTP? But again, Google is being nothing if not cautious.

But in this case, their very cautious approach is maybe easier to understand in the wake of a change that was just made in the most recent and still current Release 87 of Chrome, which is what we've got now, which they then immediately needed to roll back in an incremental update as a result of a surprising hue and cry from their users. The change was one that we covered before it happened. We were anticipating this. And that's the enforcement of secure form submission with a warning if the form's target URL is not set to an HTTPS URL.

And when we talked about it, about that this would be coming to Chrome 87, I commented at the time that this announcement surprised me because I had assumed that everyone had long ago been enforcing secure form submissions. It was years ago that we originally noted on the podcast that, even though nonsecure pages might themselves be just HTTPS, they could still be submitting their form data securely since it wasn't the URL of the submitting page, but rather the URL to which the form was sending its data as a query that really mattered.

And of course, while that was true, it was also still unadvisable to submit anything from an HTTP page since, being HTTP, anyone being able to establish a man-in-the-middle position would be able to edit the page's contents to, for example, have the page send

the form's data elsewhere. But in any event, with the release of 87, Chrome 87, Google finally joined the club of browsers that are enforcing secure form submission. And again, surprisingly late.

So what went wrong? Google apparently implemented their solution differently from everyone else's, since it began, at the release of 87, it began producing scary warnings shortly after it went live. The trouble arose because, as we know, what web pages do is generate queries; right? Web browsers query remote servers. And so the way data is submitted is kind of a kludge. They submit data in a query. And queries can either be GETs or POSTs. And in the case of a form, although you technically can send the data over a GET URL, that's considered really bad form.

So normally the URL is just some script, typically, running on the server; and the actual data which is sent is in a POST query in that query body, which contains the actual data being submitted. And of course since it's a query, everything the browser does is a query, the server then responds. And a typical response might be to return a page saying, you know, "Welcome back, Joey. You have successfully logged on," or whatever. But a more complex reply in the form of a redirect of the browser to another server URL domain might also happen. It would be just as valid. You know, the server in responding to the query can do anything it wants. And it turns out doing that, that is, sending the user somewhere else, is also somewhat common.

What Google learned the hard way was that some of these form submission redirect responses are to nonsecure HTTP URLs. And this doesn't represent any threat to the user-submitted data since the original form submission URL would be to an HTTPS URL, so the form's data will be transmitted with the encryption privacy and certificate authentication provided by HTTPS. And wherever the receiving server wants to then bounce its user is really up to it. But Chrome was originally or initially, at this launch of 87, it was being faithful to Google's HTTPS-or-die philosophy. So it freaked out when, even after the form had been submitted, the user's browser was then bounced to a nonsecure HTTP page for whatever reason.

The user would receive a warning that was designed to be scary. The warning was, you know, a big information icon, and it said: "The information you're about to submit is not secure. Because the site is using a connection that's not completely secure, your information will be visible to others." Well, first of all, that wasn't true. This would be presented after the data had been submitted, and the browser was then bounced somewhere else. So, I mean, this was really a bug in Chrome's implementation.

So anyway, upon the release of Chrome 87, Google began receiving complaints from websites that their users were suddenly receiving these bogus warnings about insecure form submissions. And it didn't take Google long to determine that it wasn't the submission itself that was triggering the error, but rather the post-submission redirect. And so shortly after Chrome 87's release, Google's software engineer Carlos Joan Rafael Ibarra Lopez stated that they are disabling the feature in Chrome 87 to "adjust" it so HTTP redirects after a secure form submission do not generate a warning. He wrote: "After considering the unexpectedly large impact this change had on form submissions that involve redirects through HTTP sites, we've decided to roll back the change for Chrome 87. We expect the configuration to be out later today, at which point it will take effect on the next Chrome restart." He says: "I'll ping this bug with updates."

He continued: "We are planning to reenable the warnings in Chrome 88, tentatively going to stable on January 19, 2021; but warning only on forms that directly submit to http://, or that redirect to http://, with the form data preserved through the redirect so it won't trigger for the cases mentioned in this bug where the http:// hop did not carry the form data. That being said," he said, "I still encourage sites to keep https:// throughout the whole redirect chain, as http:// steps will compromise user privacy by exposing the form

target location, even if no form data is being exposed." And then he says: "Apologies for the issues caused by this new warning."

So apparently everybody else who has already been warning of insecure form submission did it the right way and didn't care if the follow-on redirect was HTTP because no one else is having the problems that suddenly Chrome 87 started to trigger when Google decided to do this. And again, I'm really surprised that only now is Google doing this. The only explanation is that their metrics must have shown that too many sites were still doing this, and it would have caused too many problems. So they must have been deliberately holding back while continuing to push HTTPS everywhere until their own telemetry showed that, okay, the submissions that we are actually seeing are all to HTTPS.

So at that point they turned it on, but missed the fact from their pre-turn-on telemetry that there were redirects that were HTTP, and their implementation was actually buggy in producing a warning, a bogus warning. And in fact even in what he posted, it still sounds like they're not actually telling the truth. I mean, the idea that an HTTP redirect would carry different information than an HTTPS redirect, well, that's just not true. So they're like, okay, not wanting to completely say, yeah, well, we just screwed up the way we implemented this. But anyway, it's fixed.

This is really interesting about Firefox. As we know, the original designers of the HTTP protocol understood that having browsers download the same static content over and over and over from a remote web server would be really dumb. So from its first days, browsers employed local storage caching to effectively incrementally and conditionally move some of the remote web servers' more static data over to the users' side of the connection. And this was managed by a clean and simple mechanism that allowed any web content to indicate for how long it was allowed to be cached. And content could specify, for example, a "max age" for itself, and also a "not valid after" expiration time and date. This would cause locally cached data to self-expire.

And if a browser already had something in its cache that a new page was requesting the redisplay of, it could send a new query for that existing content with an "if modified since" header which would tell the receiving server when the cached content had been received, and the server could then check with its local copy of the content to see whether that local content had been modified since the browser had received and stored its cached copy, and then the remote server would only - it would send back a reply, I think it's a 302, not modified, meaning go ahead, there's been no change at our end, at the server end. Go ahead and immediately, instantly, redisplay from your local cache. Otherwise it would send a "200 HTTP OK" with an updated copy of that cached content. The point is caching has always been and is crucial to the performance of our web browsers.

The bad news is, unfortunately, because it inherently stores evidence of a user's browsing history, caching itself is subject to tracking and privacy abuse. And we've seen this two years ago, Spectre and Meltdown and the whole idea of things like branch prediction which could be probed for which branches were taken inside an Intel silicon. Anything that leaves any evidence of the past, we keep seeing examples of how that can be probed in order to create various sorts of vulnerabilities.

And, for example, recall that clever hack we talked about, where a tracking facility, in the case of a browser, a tracking facility would create an off-canvas link with a URL that it wanted to probe. And then it would allow that to be rendered by the browser out of sight, and then it would check the link's rendered color, knowing that the browser would color previously visited links differently than new links.

So there's another, like, anything that is happening, there is enough industry behind tracking that somebody will figure out a way to take advantage of that. So the problem is that today's browsers have been storing all of their cached content in single large pools without regard for the domain which sourced the cached item. And this has led to trackers probing the shared cache pool for its content.

The World Wide Web Consortium, the W3C, has a response to this, which is known as "client-side storage partitioning," also known as "network partitioning." And it's coming to our Firefox browsers with Release 85 of Firefox next month, in January of 2021. According to Mozilla, the following network resources are not currently partitioned, but they will be, starting with the next release of Firefox 85. Get a load of this: the HTTP cache, the image cache, the favicon cache, connection pooling, style sheet cache, DNS, HTTP authentication, speculative cache, the font cache, HSTS, OCSP, intermediate CA cache, TLS client certificates, TLS session identifiers, prefetching, preconnect, and the CORS preflight cache.

In other words, all of these things are currently being cached to improve the experience of users, and that's going to change. All of that is client-side state history data that could, unless proactively prevented, be sniffed by third-party domains, by advertisements, by third-party JavaScript libraries running in our web pages.

And while Mozilla's deployment of this new W3C standard client-side storage partitioning will be the broadest implementation of partitioning so far, turns out it's not the first. The prize for being the first goes to Apple, who began partitioning Safari's HTTP cache seven years ago, in 2013. We talked about this happening at the time. And Apple then followed through by partitioning even more of their client-side data years later that was part of their tracking prevention campaign.

Google recently partitioned Chrome's HTTP cache. That was just last month in Chrome 86. And the results were felt immediately as Google Fonts lost some of its performance when fonts could no longer be stored and be shared globally in a global HTTP cache. So unfortunately we're in this cat-and-mouse game where the technologies we're implementing just to improve the user's experience, in this case by caching content locally, is being, and there's no other way to say this other than abused, by this massive industry that wants to track users, I mean, and will expend obviously lots of time and effort to do so. I mean, to me it feels like it's unfortunate we just simply can't outlaw all of this because it is not the way these systems were intended to be used. And it's only being repurposed for abuse.

The Mozilla team has indicated that they expect to see some similar performance hits when Firefox's cache partitioning is brought online. But they are committed to taking that hit in order to improve the privacy of Firefox users. And Mozilla did say that one positive side effect of their deployment of comprehensive client-side partitioning is that Firefox 85 will finally be able to block supercookies, which are those files that abuse the shared storage medium in order to persist in browsers and allow advertisers to track user movements across the web. In creating this really strict partitioning, supercookies will just stop working in the future. So some nice mechanisms moving forward in Firefox.

One more bit of good browser news, which is that browsers are once again saying no to Kazakhstan. We've reported this, all of these efforts incrementally through the years on the podcast. And we reported some time ago that the government of Kazakhstan had been requiring their citizens to install a Government of Kazakhstan CA root certificate into each of their machines. And of course we know why. This would allow the government to perform what I would call "no complaints" man-in-the-middle interception of all web traffic of any and all of their citizens.

A Kazakhstan proxy would accept the remote connections from remote servers, decrypt, inspect, and then reencrypt the traffic under the Kazakhstan CA's identity, which would then be trusted because the Kazakhstan citizen's machine would have had installed a matching root CA. So Google, Mozilla, Apple, and Microsoft all together said, "No you don't." All four of those companies' browsers were recently updated to block any and all use of that root certificate.

A thread on Mozilla's bug reporting site first reported that this new certificate was discovered in use just over three weeks ago on December 6th. The Censored Planet website later reported that the certificate worked against dozens of web services that mostly belonged to Google, Facebook, and Twitter. So, you know, Instagram and WhatsApp and the various properties that each of those major social media sites own. And I guess, you know, you've got to give these Kazakhstan government cretins credit for trying.

Remember back in 2015 the Kazakhstan government formally applied to have their root certificate included in Mozilla's trusted root store program. But once it came to light that their entire purpose for this was to use the certificate to intercept their citizens' data, Mozilla denied the request. And then shortly afterwards the government required its citizens to manually install its certificate when it couldn't get Mozilla to do so for them, but that attempt failed after organizations took legal action.

And then, more recently, August before last, in 2019, all browsers blocked a similar attempt. And our listeners may recall the Kazakhstan government's dubious statement at the time. Reuters reported that "Kazakhstan has halted the implementation of an Internet surveillance system criticized by lawyers as illegal when the government described its initial rollout as a 'test.'" State security officials claimed they were trying to protect people in Kazakhstan from hacker attacks, online fraud, and other kinds of cyber threats. Uh-huh, right. Kazakhstan's president said in a tweet back then that he had personally ordered the test, which showed that protective measures would not inconvenience Kazakhstan Internet users. The president said, don't worry, nothing to see here. There are no grounds for concerns.

So anyway, they have tried this yet again; and, once again, they've been blocked. Back in August of 2019, we conjectured that the only feasible path for Kazakhstan, if they really insisted upon doing this, would be to create a Kazakhstan national web browser, and that then that would be the only one that would be allowed to work in-country. But of course the problem is these days it's the mobile platforms that have become dominant, and it is unlikely in the extreme that either Apple or Google or Microsoft would allow such a privacy-violating browser to be hosted in their app stores. So its use would be strictly constrained to desktops, and it's not feasible to kill the operation of mobile applications these days.

Leo: It's so hard to be an evil dictator these days. I just...

Steve: Isn't it. It really is, Leo.

Leo: It's so frustrating.

Steve: That darn technology just bites you every time.

Leo: Gosh darn it. Man.

Steve: So at this point it's Browsers 3, Kazakhstan 0. And that doesn't look like it's going to change anytime soon.

We have an interesting announcement of the newly formed Ransomware Task Force, the RTF. The newly christened Ransomware Task Force is a group of 19, I guess they're loosely related, security firms. We'll talk about them in a second. Oh, yeah, well, security firms, tech companies, and nonprofits, which include Microsoft and McAfee. Last Monday the group announced their plan to form a coalition to deal with the rising threat of ransomware. The group will focus on assessing existing technical solutions that provide protections during a ransomware attack. The RTF will commission expert papers on the topic, engage stakeholders across industries, identify gaps existing in current solutions, and then work to form a common roadmap to have issues addressed among all members.

The group's target is the creation of a standardized framework for dealing with the ransomware threat across all market segments. The single framework will be based upon an industry consensus rather than individual random advice received from single contractors. The 19 initial founding members reflect the group's commitment to building a diverse team of experts. We've got Aspen Digital that describes itself as a policy maker group; Citrix, of course, the equipment vendor; the Cyber Threat Alliance, which is a cybersecurity industry sharing group; Cybereason, a security firm; the CyberPeace Institute, which is a nonprofit dedicated to help victims of cyberattacks; the Cybersecurity Coalition, another policy maker group; the Global Cyber Alliance, a non-profit dedicated to reducing cyber risks; the Institute for Security and Technology, a policy making group; McAfee, of course; Microsoft; Rapid7. Resilience, who are a cyber insurance provider, you can imagine they would like some help in this industry.

We've got the SecurityScorecard, a compliance and risk management organization; the Shadowserver Foundation, a non-profit security organization; Stratigos Security, cybersecurity consulting firm; Team Cymru, which is, we know, we speak of them often, a threat intelligence firm. The Third Way is a think tank; the UT Austin Strauss Center, a research group; and the Venable LLP, a law firm. So this Ransomware Task Force website, including full membership details and leadership roles, will be launched next month, in January of '21. So I am sure that we'll be covering that. And that's expected to be followed by a two- to three-month sprint to get the taskforce up and running and off the ground. So it's going to be interesting to see how this develops and what they may be able to produce. And it's just nice to see people stepping up, recognizing that we've got a problem here that could use some industry-wide consensus putting our heads together.

So this week in WordPress. We have once again more than five million WordPress sites in critical danger thanks to their use of a popular plugin called Contact Form 7, which looks to be a nice outfit. As I dug into this a little bit, I was impressed by what I saw from Contact Form 7. If nothing else they are being responsible. In this case, the trouble arises from a lack of sufficient filename sanitization in the plugin's file upload filter. This allows a file of the form, you know, anything, xyz.php\t.jpg to be seen by the upload filter as a benign JPEG image file. What could possibly go wrong? Whereas it will be seen by the PHP interpreter as a valid PHP script, the trick being to use a so-called "double extension," you know, .php\t.jpg, and different components of the system parse that double extension filename differently, which therein causes the problem.

In this case the flaw has a CVE designation of 2020-35489. Yes, we've got your CVEs. Get 'em right here, 35,489. And Astra Security, during a security audit for a client whose WordPress site was using the Contact Form 7 plugin, is where this was found. A representative from Astra Security said: "Seeing the criticality of the vulnerability and the

number of WordPress websites" - more than 5 million - "using this popular plugin, we quickly reported the vulnerability. The developer was even quicker in issuing a fix."

As I said, impressed by Contact Form 7, whose domain is ContactForm7.com. They didn't even bury this down in some obscure URL. If you go to ContactForm7.com, right at the top of the home page it says: "Contact Form 7 5.3.2 has been released. This is an urgent security and maintenance release. We strongly encourage you to update to it immediately." And they said: "An unrestricted file upload vulnerability has been found in Contact Form 7 5.3.1 and older versions. Utilizing this vulnerability, a form submitter can bypass Contact Form 7's filename sanitization, and upload a file which can be executed as a script on the host server. So props to them for stepping right up and alerting everyone.

And it was interesting, I was curious because this was a chronological listing of postings. So I didn't have to scroll down very far, only down to August 24th of this year, so in the late summer, where I found "Heads-up about Auto-Updates." They posted: "WordPress 5.5 has introduced the auto-update feature for plugins and themes. Keeping plugins," they wrote, "and themes updated to the latest version is a key factor in managing your WordPress site securely. We strongly recommend you enable auto-updates for the Contact Form 7 plugin, but you should also be aware that there are risks involved in the use of auto-updates.

They said: "In the following cases, consider disabling auto-updates and doing an update manually." And there's three bullet points. First, you use plugins that extend the functionality of Contact Form 7, that is to say, an add-on plugin. So in this case I guess plugins can have plugins. The second, you use a theme that overrides the CSS style rules of Contact Form 7; or, three, you apply coding customization of some sort to Contact Form 7.

They said: "In those cases, updating Contact Form 7 or one of the plugins or themes that affect Contact Form 7 might bring about incompatibility risks between them. And if you do it automatically, you might not even realize problems are occurring on the site." They said: "Managing your site's security is your responsibility. Update your plugins and themes in a proper way. If there is a plugin or theme that is an obstacle to updating other parts, you should make a decision to remove it."

So again, here we have a problem. The fundamental problem is that well-meaning amateurs are able to create compelling plugins for WordPress which take the base WordPress system, whose security is pretty good, and completely wreck it. The plugin authors aren't doing it on purpose. But they're just not security expert PHP authors. And it turns out there are all kinds of ways to get around well-meaning PHP scripts. And this is like a weekly theme now for Security Now!, is how many tens of millions of WordPress sites are vulnerable to the most recently discovered problem.

So, boy. If you're going to run, if you're going to host a WordPress site - and first off, don't. But if you want to, the only way to do it is to really sequester a machine, maybe make it a VM if you don't have lots of hardware, really sequester it so that it has its own private SQL database. Don't reach out and go touch like a common SQL store where you have everything else in your enterprise. Give it its own local SQL instance on that virtual machine, and firmly firewall it so that it only has access to send and receive web, email, and DNS traffic. And do your best to just really firewall and sandbox that thing because all the evidence - oh, and don't install plugins. If you can in any way resist the temptation from, oh, this looks great, you know, whoa. Just really try not to. So wow.

It's disturbing when really widespread problems come back again. And in this case we have the U.S. cybersecurity infrastructure and security agency that we're hearing more and more about. It's a division of the DHS, the Department of Homeland Security. This is

CISA that has been very active in talking about the amazing SolarWinds attacks. There's been lots of production of warnings and things.

In this case, they have warned of newly discovered critical vulnerabilities in the low-level TCP/IP software library developed by Treck that, if weaponized, could allow remote attackers to run arbitrary commands and mount denial-of-service attacks. The specific four flaws affect Treck's widely used TCP/IP stack v6.0.1.67 and earlier, and were reported to Treck by Intel. Yes, because Intel is one of the users of this widely used, IoT-embedded TCP/IP stack. And you can just imagine where Intel has this thing.

Two of these four are rated critical in severity, and here's the problem. Treck's embedded TCP/IP stack is deployed worldwide in manufacturing, information technology, healthcare, transportation systems, and elsewhere. The most severe of the four is a heap-based buffer overflow in the Treck HTTP Server component that could permit an adversary to crash/reset the target device and even execute remote code. It carries a CVSS score of 9.8 out of a maximum of 10. And of course we know what that means. Being in the HTTP server, any embedded device or IoT gizmo that exposes an HTTP service onto the public Internet, whether or not it provides strong access authentication, could nevertheless be compromised remotely.

The second flaw is an out-of-bounds write in the IPv6 component, which received a CVSS score of 9.1, so not 9.8, but still it's critical because it could be exploited by an unauthenticated user to cause a denial-of-service condition via network access. So not exploiting remote code, but crashing the system remotely. But if that system were a critical industrial control system, controlling something and needing to stay up, crashing it could be bad. The two other vulnerabilities are an out-of-bounds read in the IPv6 component that could be leveraged by an unauthenticated attacker to cause a denial of service and an improper input validation in the same module that could result in an out-of-bounds read of up to three bytes via network access. Okay, again, those both had much lower CVSSes.

Now, this next part I really like. Get a load of this. The CISA's disclosure said: "Treck recommends users to update the stack to v6.0.1.68 to address the flaws. In cases where the latest patches cannot be applied, it's advised that firewall rules are implemented to filter out packets that contain a negative content length in the HTTP header." Yeah. I love that. So the hacks involve sending HTTP queries containing a negative content length. So there is presumably code that's checking for a content length that's greater than some value.

But as we know, in two's-complement binary encoding, a signed negative value will appear as a very large unsigned positive value when it's interpreted by logic that's expecting to receive an unsigned length. So a typical signed/unsigned type confusion problem. Presumably somewhere they're doing a signed comparison. But if you put in a negative value, like a minus sign in front of it, the interpretation code blows up and allows you to compromise the stack.

And here's the most daunting bit. If the name Treck and the idea of disastrous TCP/IP stacks might be ringing some bells, it did when I ran across it, that would be because we first encountered these guys this past summer in June with the so-called Ripple 20 attacks, which were, remember, 19 different horrible security problems in Treck's massively widely used embedded TCP/IP networking products. They named the large set of vulnerabilities "Ripple 20" due to the ripple effects that are inherent when problems exist at one end of a long supply chain. The potential devastation will ripple out to affect hundreds of millions of devices.

And in fact, at the time it was estimated that Treck had, what was it, 105 different customers, and that these vulnerabilities probably affected more than a billion devices.

So to those 20, or actually 19, we can now add four new ones that are obviously trivial to exploit if you do your homework. And in this case Treck's customers range from one-person boutique shops to Fortune 500 multinational corporations, including HP, Schneider Electric, Intel, Rockwell Automation, Caterpillar, Baxter, as well as many other major international vendors in medical, transportation, industrial control, enterprise, energy, oil and gas, telecom, retail, commerce, and other industries.

And, you know, we talked about this last summer, as I said, when these first 19 flaws were brought to our attention. But it's made, you could argue, even more chilling today in the wake of the massive SolarWinds Sunburst intrusions. It's one thing to know intellectually that probably most of our IoT devices are highly vulnerable to remote attack and takeover. But what really puts a sharp point on that is the idea that we now know without any question that there are entities bearing us malice who also have the capability to turn these theoretical vulnerabilities into fully practical attacks.

At this point, because all of these IoT devices already have left the gate - I mean, they're installed. There are billions of them - cameras, webcams, security cameras, home alarm systems, doorbells, I mean, just everything. They're out there. And they're all on the Internet. They all have TCP/IP stacks. It truly is chilling that this is happening. And, you know, the next podcast is going to be 800. That leaves us with 200, or actually 199.

Leo: Don't remind me. Don't remind me. We've got 200 left today.

Steve: Yes. So I think we're still going to be having some interesting podcasts before the curtain drops on this.

Leo: But it'll all be fixed by 999; right? After that it's all over.

Steve: Oh, exactly, Leo. We're going to scrap all of these first-generation IoT devices, and we're not going to have any more problems.

Leo: Kazakhstan's going to have a root certificate, and all's right with the world. It's just a matter of time, 200 episodes.

Steve: Wow. Wow.

Leo: Wow is right.

Steve: I have a bit of closing-the-loop feedback. Anthony Lipke tweeted. To @SGgrc, he said: "I remain a fan of Newgrounds, a flash site." And Leo, you ought to put in www.newgrounds.com.

Leo: Dare I?

Steve: Just to see what it looks like.

Leo: I've been getting calls like this all month from people who say, oh, no, my favorite site's on Flash.

Steve: Yeah. This looks like the Revenge of Anime and eight-bit pixel stuff. Anyway, Anthony says: "I haven't seen something take that place for games and animation. That said, they're part of great efforts to preserve the content. You're likely already aware, but it seems worth mentioning besides just saying Flash is dead." And, you know, I am sympathetic. I mean, holding on, preserving Internet history and culture is kind of cool. One could argue that the GRC.com website is inherently preserving Internet history because, yes, a lot of my graphics are blocky like eight-bits.

Leo: Well, I should point - he'll get onto his thing. But the Internet Archive has done - there's a Flash emulator, and they've done a great job, because that's their job, of preserving some of the stupidest Flash sites you have ever seen. But it's part of our, you know, "Peanut Butter Jelly Time" is part of our culture. So I agree with you 100%. But the way to do this is not to keep sites like that alive, or to try to somehow save Flash. By no means. Let's let time move on. But acknowledge that these were important in their day, yeah. I completely agree.

Steve: Right, right. So I did a little bit of digging around, and I discovered, because there is a concern out on the Internet, like okay, we need to still have Flash running.

Leo: No.

Steve: So what can you do?

Leo: No, we don't. Stop it.

Steve: Well, certainly not in our browsers. It was the browser. It was the embedded Flash Player. I mean, but Leo, I also agree with you. I mean, it's like, yes, we have HTML5.

Leo: Yeah. You don't need AmigaDOS. You don't need any, you know, this stuff's over. We don't need it anymore.

Steve: Yes. For those who are unwilling to move on, you can still get a standalone Adobe Flash Player if you google "flash player projector content debugger." I've got the link to it in the show notes. It is available, standalone, does not run in a browser, for Windows, Mac, and Linux, being published by Adobe. My guess is you probably ought to get it soon because maybe Adobe, when they sunset Flash, like maybe in two days, on January 1st, this may go away. But it's there now.

And then the other thing is there's something called BlueMaxima.org, and they have a Flashpoint web game preservation project. And they said: "Internet history and culture is important, and content made on web platforms including, but not limited to Adobe Flash, make up a significant portion of that culture." Okay, well, I guess that's debatable. They said: "This project is dedicated to preserving as many experiences from these platforms

as possible, so that they aren't lost to time. Since early 2018, Flashpoint has saved more than 70,000 games and 8,000 animations running on 20 different platforms.

"Flashpoint was started in January 2018 by BlueMaxima in an attempt to outrun the disappearance of content prior to the death of Flash. It has since evolved into an international project involving over 100 community contributors, encompassing both web games and animations created for numerous Internet plugins, frameworks, and standards." So BlueMaxima.org/flashpoint.

Leo: Yeah. This is an emulator. It's kind of like a Nintendo emulator or something like that. That's fine.

Steve: Yup. Yup. So I just wanted to give Anthony his due and to say, yes, we do acknowledge that there is valuable historical culture and blocky eight-bit graphics so you can still be there if you want to.

Leo: There's also a WebAssem Flash emulator, and there's one written in Rust called Ruffle. This isn't going to go away. Emulating Flash is easy. Well, relatively. But please, let's not try to save these sites online. Let's get rid of them now.

Steve: And so I can announce...

Leo: Uh-oh.

Steve: ...that ReadSpeed is ready.

Leo: Yes.

Steve: Although this particular timing was never my plan, the first release of ReadSpeed Benchmark went widely public on Christmas Eve. I tweeted it to my Twitter audience, and I am now formally inviting this podcast audience to give it a try. The code had settled and had been stable for quite some time. And I'd had time to get ReadSpeed's home page at GRC ready. I annotated a sample run from my own large multi-drive test system, and I made an 11-minute video walkthrough of that Benchmark running with my own voiceover commentary highlighting the various events of interest which occur during that Benchmark. So I thought it's finally time to let the world have a crack at it. It's on the freeware page. It's under Freeware Utilities ReadSpeed. Or if you just google "GRC ReadSpeed" you'll find it.

And, interestingly, in parallel, there's been a lot of research work going on over in the SpinRite development newsgroup regarding SpinRite's ability to improve SSD performance. So check out this before-and-after benchmarking of an OCZ Vertex 3 SSD. One of the guys just posted this yesterday. The Benchmark normally just summarizes the performance of a 1GB read at five different locations. But you can add a /1, /2, /3, or /4 to request the display of greater and greater levels of granularity. The cool thing is in this first chart you'll see that the 0% point of this OCZ Vertex 3, this is an SSD, it was reading that first gig at 300MB per second. The second, at the 25% point, it was 320MB per second. And at the middle, 324.

If you look then down at the numbers below the 0% line, you'll see that they're not really good. There's a 177.9, then an 89.0. These are equivalent megabytes per second for the individual pieces, individual requests. But now look at the second chart. This is after running a Level 3, a SpinRite Level 3 on that OCZ Vertex 3. We went from 300MB per second to 354, 320 to 353, 324 to 352 and the rest. And if you look at the same detail breakdown, it's all been repaired. It's up to 360.

Leo: Is that from trimming? Or what is it doing?

Steve: No, because SpinRite is trimming naive. It doesn't know about trimming yet. That is, what we are seeing is that these big slowdowns are caused by bit errors which are requiring error correction code. And so SpinRite in rewriting these regions is fixing the sectors. It is exactly like it used to be in the days of hard...

Leo: On spinning drives, yeah.

Steve: Yes. Exactly like in the days of spinning drives. So we're definitely seeing that SpinRite - we already knew that SpinRite could fix SSDs. We've been sharing reports of those anecdotally for years of, like, my SSD died. I "spunrote" it, and now it works again. It's like, oh, okay, I guess that there will be a future for SpinRite. Well, it's going to be even brighter than we thought because in the future SpinRite will be looking at the timing of the scan it does. It will first do a characterization pass to learn about the performance of the SSD, and then go back and do spot repair of all the areas which are clearly in trouble of future failure and bring them back up to speed in the process.

So anyway, it's looking like we're going to be having a lot of fun for a number of years to come with SpinRite. And there you're showing the annotated results of the Benchmark on my machine. I then have in text some of the things that I found that were there. And then that page has an 11-minute video that shows the Benchmark actually running as it goes.

Leo: I just wish I could run this on a Mac.

Steve: Yup.

Leo: Nope. That requires DOS, and that's life, yeah.

Steve: Also, just one other little last bit is that last week I announced InitDisk 4's release. We ran across a weird problem. We were having a situation, there were some anecdotal reports that InitDisk would sometimes produce a write error. It turns out that someone in the SpinRite dev group had a USB stick that was producing that error. So I got him to send me an image of that stick, and I was able to recreate the problem. It only happened under Win10, not under Win7. So it's something that Microsoft changed in Windows 10.

What I learned was that, if you were trying to run InitDisk - or for that matter ReadSpeed because they share the same technology, as will SpinRite and Beyond Recall. If you were trying to reformat a USB drive under Windows 10, again, not other Windows, which had not been partitioned, because InitDisk is putting a partition sector down

because we found that gives you better compatibility over time, Windows 10 would just freak out. So I fixed that. There is now InitDisk Release 5, and that was the last thing I needed to do before ReadSpeed was ready because, as I said, it inherits the same technology.

So lots of cool stuff coming out as Christmas presents for everybody. And again, if ReadSpeed works for you, then so will SpinRite 6.1. It uses the same technology. It is basically a proof of technology for the new drivers that I will then, now, starting in 2021, shortly be moving over into SpinRite in order to create 6.1. So lots of fun holiday news on that front.

Okay. So we wrap with everything we've learned since our podcast two weeks ago and since the three weeks ago surprise disclosure by FireEye of the SolarWinds intrusion. The malware that was first found was given the name Sunburst. We now have a second piece of malware. It's been named Supernova. And it should not surprise anyone that more intelligence is being continually uncovered about this event, which is being called the biggest computer hack in history. I'm sure all of us watching any television, any news shows over the last couple weeks, it was like, "the big hack." And, I mean, it was a big deal. And of course we should mention, Leo, that we had a great TWiT Sunday show.

Leo: Yes. People should watch that.

Steve: So any of our listeners, yes. Any of our listeners, we got a lot of positive feedback about the "old guy" show on TWiT on Sunday.

Leo: Yeah, that was the December 20th version. We did talk a lot about SolarWinds. But it's an ongoing story. I also feel like it's one that nobody knows what to do with because it doesn't seem like there's much we can do with it.

Steve: No. So one thing I want to clear up first was that two weeks ago I said that SolarWinds Orion was an appliance. That was incorrect. It's just a software system that can be loaded and run on any qualifying Windows system. So I wanted to correct that for the record. Not an appliance, it's just software. And to my mind, the biggest revelation since the initial discovery of that Orion.dll being hacked, signed, and then delivered to approximately 18,000 SolarWinds customers via a software update is now that compelling evidence has been found of a second entirely separate backdoor in SolarWinds offerings. The evidence leads forensic investigators to believe that this second, now it's being named the Supernova backdoor, was planted by a second threat actor. And at this point, if you had any equity stake in SolarWinds, you're probably not happy because they've taken a serious reputation hit.

The second piece of so-called Supernova malware is an extremely sophisticated web shell which was also planted in the code of Orion's network and their applications monitoring platform. It enabled the attackers to run arbitrary code on any of the machines hosting the trojanized version of the Orion software. The web shell, it's a modified version of a legitimate .NET library DLL named "app_web_logoimagehandler.ashx dot" and then a serial number "b6031896.dll" that's present in SolarWinds Orion software. And that DLL was very cleverly designed to get its nefarious job done while proactively evading automated defense mechanisms.

So again, this shows a high-level adversary, and it is believed a different high-level adversary. So by design, the Orion software uses this DLL to expose an HTTP API which allows the host to respond to other subsystems when querying for a specific GIF image.

In his technical report, published on the 17th, Matt Tennis, a senior staff security researcher at Unit 42 of Palo Alto Networks, wrote that the malware could slip past even careful manual analysis since the code implemented in the legitimate DLL is innocuous and is of relatively high quality. From Matt's introduction to this second threat, we also learned something significant about the attackers that hasn't been widely reported.

Matt wrote: "The actors behind the supply chain attack on SolarWinds' Orion software have demonstrated a high degree of technical sophistication and attention to operational security, as well as a novel combination of techniques in the potential compromise of approximately 18,000 SolarWinds customers. As published in the original disclosure, the attackers were observed removing their initial backdoor once a more legitimate method of persistence was obtained." In other words, the attackers used their initial backdoor intrusion mechanism only until they were able to obtain keys to the front door, at which point the backdoor mechanism was shut down and went quiescent to avoid any chance of its detection.

In describing this second Supernova backdoor, Matt writes: "The analysis shows that the threat actor added into the legitimate SolarWinds file four new parameters to receive signals from the command-and-control infrastructure. The malicious code contains only one method, DynamicRun, which compiles on the fly the parameters into a .NET assembly in memory, thus leaving no artifacts on the disk of a compromised device. This way, the attacker can send arbitrary code to the infected device and run it in the context of the user, who most of the time will have high privileges and visibility on the network. It is unclear," he wrote, "how long Supernova has been in the Orion software, but a malware analysis system shows a compilation timestamp of March 24th, 2020."

And when I saw that, I thought, what? The timing of that may seem to be very close to the March 6th, 2020 signing of the first instance of the original Sunburst backdoor. But we have also learned that the initial intrusion by the first actor into SolarWinds was likely made back in the previous October 2019. And that's news. Okay. Let me say that again. We have since learned that the initial intrusion that the first actor into SolarWinds made was likely the previous October 2019. What was discovered was a benign do-nothing change was found in the SolarWinds source code base dating back to October 2019. It literally does nothing. I have a picture of the decompiled source that was produced by a JetBrains decompiler showing that this call that was added does nothing but return.

Leo: That's interesting.

Steve: It literally does nothing.

Leo: Why would you put that in?

Steve: The researchers conjecture that the change was injected just to allow them to determine whether this change would, A, go undetected; and, B, eventually be disseminated out into SolarWinds' global customer base.

Leo: This is just a test.

Steve: Yes. So the bad guys were SolarWinds customers. Obviously they had access to all of this stuff in order to reverse engineer it. So they infected the codebase back in October of 2019. Didn't make a change that did anything. But they wanted to see

whether a subsequent update that they would receive from SolarWinds would contain that modification.

Leo: And be signed and get through any security.

Steve: Yes, yes, yes.

Leo: By the way, this is exactly what the Russian - what Fancy and Cozy Bear do. They're very cautious and careful and step-by-step. And it seems to be they often do test runs of their hacks.

Steve: Yes.

Leo: I mean, I'm sure there are many fingerprints that lead people to think it's - I think it was Cozy Bear they thought it was.

Steve: Yes.

Leo: But that's, I bet, one of them. I mean, what hacker does that? Let's see, let's make sure it works before we go too far.

Steve: Well, and what, October? It's like that's six months that they waited.

Leo: Yeah, very patient, yeah.

Steve: So, yeah, they're in for the long term.

Leo: That's a nation-state hacker, not somebody trying to get ransomware over the top, yeah.

Steve: Yup, exactly. Matt says that this second intrusion was based upon the findings, that is, this Supernova bears the hallmarks of an advanced hacking group that took compromise via a web shell to a new level. He wrote: "Although .NET web shells are fairly common, most publicly researched samples ingest command-and-control parameters and perform some relatively surface-level exploitation." In other words, he said that taking a valid .NET program as a parameter, that is, you're taking the program, they're feeding the program into this modified HTTP as a query parameter, which then performs an on-the-fly compilation assembly so that the remotely provided code is then running in memory, makes Supernova a rare encounter, as it eliminates the need for additional network callbacks aside from the initial command-and-control request. Most web shells run their payloads in the context of the runtime environment or by calling a subshell or process such as CMD or PowerShell or Bash.

Microsoft believes that this Supernova web shell is likely the creation of a different adversary than the one that was first discovered by FireEye. Microsoft wrote: "In an

interesting turn of events, the investigation of the whole SolarWinds compromise led to the discovery of an additional malware." Now, you can imagine that these other guys are not happy because they engineered this different way and were apparently operating unseen for who knows how long. I mean, remember that that was, what, March 6th? So this second group has also been in and mucking around since March 6th. So they're not happy that the first group got caught because, as Microsoft says, this led to the discovery of an additional malware that also affects the SolarWinds Orion product, but has been determined to be likely unrelated to this compromise and used by a different threat actor.

Leo: Oh, that's such bad news. Oh, my god.

Steve: I know.

Leo: It's Swiss cheese.

Steve: Yeah, it's a mess, Leo. And one argument for this theory is that, unlike the Sunburst DLL that slipped into SolarWinds source code repository and that was validly signed, Supernova does not carry a digital signature, suggesting that that second group were not in a position to do that.

Kim Zetter, writing for Yahoo! News, added some good detail to the saga. He wrote: "Hackers who breached federal agency networks through software made by a company called SolarWinds appear to have conducted a test run of their broad espionage campaign last year, according to sources with knowledge of the operation. The hackers distributed malicious files from the SolarWinds network in October of 2019, five months before previously reported files were sent to victims through the company's software update servers. The October files, distributed to customers on October 10th, did not have a backdoor embedded in them, however, in the way that subsequent malicious files that victims downloaded in the spring of 2020 did, and these files went undetected until this month.

"A source familiar with the investigation told Yahoo News: 'We're thinking they wanted to test whether or not it was going to work and whether it would be detected. So it was more or less a dry run. They took their time. They decided to not go out with an actual backdoor right away. That signifies that they're a little bit more disciplined'" - or I would say a lot more disciplined - "'and deliberate.'"

Okay. One other little interesting bit is the original Sunburst malware, which was discovered by FireEye, used a Domain name Generation Algorithm to obscure the lookups that it was doing to find its command-and-control server. And let's not forget that this was successful until whatever it was that FireEye discovered. The cool thing is, as part of its detection avoidance system, the malware incorporated its own internal kill switch, that is, the malware had a kill switch. So working together, Microsoft, FireEye, and GoDaddy have decided to cause the malware to shut itself down.

After the obscure domain name is generated as a subdomain of avsvmcloud.com, a DNS Address record lookup is performed by the malware. The resolved address is checked against a hard-coded list of networks, basically to determine whether it is being probed, one of which happens to belong to Microsoft. And if the IP address matches any of those networks, the malware will update a configuration key named "ReportWatcherRetry" to prevent its own further execution and will then terminate itself permanently.

So of course the reason this is done is that malware wants to detect when it is being run in a sandbox, right, when researchers have found a copy of it and are working to see what it does. So one of the things that researchers will typically do is they will see that it's making DNS queries, and they will grab those and change them to a non-routable IP. So sure enough, the IP kill switch list starts off with 10.0.0.0/8; 172.16.0.0/12; 192.168.0.0/16; 224.0.0.0/3; some IPv6 IPs; and then 20.140.0.0/15, which is a Microsoft IP block. And there are three others.

So in working with GoDaddy, they arranged to have the avsvmcloud.com return this Microsoft IP, which could then, across the entire world, no matter where the malware was, it would see that, freak out, flip the ReportWatcherRetry switch, and then terminate itself in order to go quiescent. But also note that the IP address querying for the domain's A record can be obtained, which is where GoDaddy comes in, to reveal the IP of the local DNS server resolving for any still-active malware. Thus it's possible to determine who has any still-active intrusion.

So GoDaddy was the DNS provider. They began collecting their records and finding all the IPs of anybody making those wacky queries to avsvmcloud.com. That gave them a list of the DNS servers that were servicing the malware, and that allowed them to determine which organizations were infected.

There was another piece of this, also. As we've often noted, of course, once a bad guy has been crawling around inside a network, especially a huge and complex network, and especially a highly skilled bad guy, it's never really possible - and this is to your point, Leo - to know that everything that they may have done while they were inside there has been found and reversed. Remember, it's quite possible for malware to take up residence inside a printer, you know, we've talked about some years ago that did that, or a security camera, or pretty much anything else these days.

FireEye was quoted by the tech press, saying: "In the intrusions FireEye has seen, this actor moved quickly to establish additional persistent mechanisms for access to victim networks beyond the Sunburst backdoor. This kill switch will not remove the actor from victim networks where they have established other backdoors." Meaning, yeah, sure, we've shut down the way they got in. But one of the things we've seen them do is immediately establish backup ways to remain in after that backdoor has been shut down.

The U.S.'s CISA followed up, talked about the ongoing discovery of the breadth and the depth of this attack, saying: "This APT actor" - Advanced Persistent Threat actor - "has demonstrated patience, operational security, and complex tradecraft in these intrusions. CISA expects that removing this threat actor from compromised environments will be highly complex and challenging for organizations."

And one final point is that the obfuscation used by the domain name generator has been reverse engineered.

Leo: Oh, good.

Steve: And logs, yes, logs. Scroll down and look at the image, Leo, on the last page of the show notes. Logs of previous DNS queries have been obtained and decoded. This resulted in the discovery of many additional infected corporations who have not yet gone public with any disclosures. And a bunch of them are quite tasty. The biggest names on the list include the likes of Cisco, SAP, Intel, Cox Communications, Deloitte, Nvidia, Fujitsu, Belkin, Amerisafe, Lukoil, Rakuten, Check Point, Optimizely, Digital Reach, Digital Sense, and probably MediaTek. So in the show notes we have a snapshot picture of some of them that have been decoded. So we've heard a lot about the various public

organizations that have been attacked, and government organizations. But, boy, you can imagine there were some Christmases that were ruined by this disclosure in early December.

Leo: No kidding. So they now think there are two different, unrelated threat actors?

Steve: Yes.

Leo: That's even worse.

Steve: Yes. Yes.

Leo: Because, I mean, that's the two we found.

Steve: Yes. And the code of this web shell, the second one, as I mentioned, there is, I mean, it's sort of sad, too, because this DLL allows other components of Orion to query for a GIF, like to put a logo on the user interface. That's what it does. And so they slightly modified the HTTP server that this DLL runs to allow it to accept .NET source code...

Leo: Terrible.

Steve: ...as a query parameter.

Leo: Terrible.

Steve: So it then compiles whatever the attacker wants and runs it with the system privileges that this system is running under, not forking out a shell which would then be running under the user's privileges. So it is a devastatingly clever hack and attack, and arguably worse than the first one we learned about.

Leo: Yeah. And yet I think the takeaway is...

Steve: If there's two...

Leo: If two, there's more, and it was too damn easy. So we're kind of in trouble. I mean, I think maybe the notion of a secure system has to be rethought. You start from scratch.

Steve: It's an oxymoron, Leo.

Leo: It's an oxymoron. It's really terrible.

Steve: I know. It's disheartening.

Leo: I don't know. I don't know. At this point you just - it's like privacy in general. It's like, well, it's over. So now what? Security is history. And that's why I think people feel like, well, what do we do? It's not like...

Steve: Yeah. And what's really interesting is that they have stayed a step ahead because we went from don't let bad guys in to then saying, okay, well, maybe we can't keep them out, so we're going to have intrusion detection systems to detect them when they're in. But they deliberately designed their traffic to look like valid traffic. So the IDS that's like inspecting everything would go, yeah, nothing to see. Everything looks fine. Meanwhile, .NET code is going by and is being compiled in RAM.

Leo: Yeah. But it's also hard to think what the end game is because, all right, they're in there. Now what are you going to do? I mean, is it industrial espionage? Is it military espionage? I would hope that at this point people in the military, for instance, are saying, okay, well, it's now back to zero-trust architectures. We have to do a better job. I don't know what the answer is.

Steve: Well, and it's why I've been preaching network segmentation now for quite a while.

Leo: Yeah. That's key. That's key.

Steve: When it became so clear that we really cannot trust our IoT devices.

Leo: Right.

Steve: They just have to be running on an isolated subnetwork within enterprises and within our own homes for peace of mind. And it's a fun project for the holidays for our listeners.

Leo: Yeah. Steve Gibson, I'm so glad we didn't take this last show of the year off. It's always important to get every bit of security information we can get. And there's no better guy to get it from than Steve. If you want to take a look at his new Benchmark, if you want to get a copy of the current version of SpinRite and be in line to get the next version the minute it comes out, it's all at GRC.com. That's his website, the Gibson Research Corporation.

While you're there, you might want to check out the show because he puts 16Kb versions of the show, 64Kb versions, and very nicely human-written transcriptions up at GRC.com. So you can get the show there. You can get it from us at TWiT.tv/sn for Security Now!. There's a YouTube channel. You could subscribe there. Or you could just get your favorite podcast application and subscribe and pick - we have video as well as audio - pick the version you like, subscribe. You probably not only don't want to miss an episode, but I'm thinking you want to collect them and keep

them because this is kind of an archive of the history of the world in the 21st Century that's just fascinating.

Steve: From Honey Monkeys on.

Leo: Yeah, I mean, it's just fascinating. And this SolarWinds thing is kind of the icing on the cake, I mean, not in a good way, but wow. We do the show on Tuesdays, right after MacBreak Weekly. That's about 1:30 Pacific, 4:30 Eastern time on Tuesdays, 21:30 UTC. You can watch us do it live at TWiT.tv/live. There's audio and video streams there. Chat with us at irc.twit.tv. Steve takes messages, direct messages. He's open to all on Twitter, @SGgrc. Or GRC.com/feedback if you've got questions, thoughts, comments. And otherwise we'll see you next week because that's a brand new year.

Steve: And I'll also just mention, I forgot to mention that, if anyone just wants to take sort of a passive look, the forums.grc.com, the top of them are a set of forums, ReadSpeed Benchmark topics, where you'll find Booting DOS, Running ReadSpeed, ReadSpeed Results, ReadSpeed Problems, and ReadSpeed Release History.

Leo: Nice, nice.

Steve: But just scrolling through ReadSpeed Results you can immediately get a sense for what people are seeing, and lots of interesting dialogue there. The forums have really taken off. And of course I built them to allow us to have a public means of exchanging conversation about this work, and then SpinRite next.

Leo: Nice.

Steve: And I can't wait to get started on SpinRite in 2021.

Leo: Nice. Very nice. Yup, 2021, January 5th there'll be Episode 800.

Steve: Yay.

Leo: We begin our last couple of hundred episodes.

Steve: The countdown.

Leo: The countdown. Thanks, Steve. Have a great week, and I'll see you next time on Security Now!.

Steve: Thanks, buddy.