SECURITY NOW!

Transcript of Episode #784

BlindSide & BLURtooth

Description: This week we look at the Chrome browser's proactive technology which is designed to punish abusive ads. We also look at the last hurrah for exploiting IE and Adobe Flash users, some Microsoft Edge updates, last Tuesday's Microsoft Patch-a-Palooza, Zoom's new implementation of two- factor authentication, that very bad WordPress File Manager attack two weeks out, the new Raccoon attack against TLS, and a quick SpinRite update. Then we conclude with a look at two newly discovered attacks named BlindSide and BLURtooth.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-784.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-784-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Lots to talk about. What Google's going to do about abusive ads. Turns out they're going to block them. One last hurrah for IE and Flash exploits. And then Steve will talk about a very interesting new attack on servers, actually two of them, called BlindSide and BLURtooth. Security Now! is next.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 784, recorded Tuesday, September 15th, 2020: BlindSide and BLURtooth.

It's time for Security Now!, the show where we cover your security and privacy online with Steve Gibson, the man in charge of Security Now! from GRC.com. Oh, it's spooky, spooky.

Steve Gibson: It is.

Leo: This is not our Halloween episode.

Steve: I think I've had too much caffeine.

Leo: No, not possible.

Steve: So this was going to be named "BlindSide, BLURtooth, and Raccoon." But as I was initially looking through the news of the week I thought, okay, you know, I was a little worried that it wouldn't fit on the screen. And as it turns out, Raccoon didn't make the cut. We'll talk about it because it is something that happened, but it isn't deserving of any particular depth. So it's just BlindSide and BLURtooth, but both are really interesting.

Leo: Have exploits, since we started doing the show 15 years ago, have the names gotten cuter? Or is it just my imagination?

Steve: Oh, they're definitely gotten their own logos and websites.

Leo: Yeah.

Steve: We didn't have that before. So, yeah. And no, I think you're right, I think there's...

Leo: We've come a long way from Melissa; you know?

Steve: Yeah, and Honey Monkey, that sort of stirs up a strange image. So anyway, yes. We're going to take a look at the Chrome browser's proactive technology, which is designed to punish abusive ads. We're also going to look at the last hurrah for exploiting IE and Adobe Flash users before those finally, thank goodness, fade away into oblivion and history. We've got some Microsoft Edge updates which are interesting, so I like to sort of do browser stuff at the top of the show since that's become its own category because it's what we expose ourselves to when we venture out on the Internet.

We've also got last Tuesday's what you'd have to call in retrospect, looking over what happened, a Patch-a-Palooza, which we'll discuss. We've got Zoom's new implementation of two-factor authentication, a follow-up on that very bad WordPress File Manager attack, now two weeks old, and there's been some updates. We've also got, as I mentioned, a new, it's called the Raccoon attack, against TLS. Also a quick SpinRite update to sort of keep our listeners current with what's going on there. And then we'll conclude with a deeper look at two newly discovered attacks named BlindSide and BLURtooth. So I think another fun...

Leo: Sounds like a CBS cop show.

Steve: Another fun couple hours for our listeners.

Leo: BlindSide and BLURtooth. It should be fun. On we go. Do you have a picture? Oh, yes, you do.

Steve: Oh, yes, I do.

Leo: Oh, yes, we do.

Steve: This came up a couple weeks ago on Firefox, and I hadn't ever seen it before, so I thought that was interesting. It was a message when I tried to go - actually, I went to www.coolmagnetman.com.

Leo: Okay, good. Yeah, magnets are cool, I agree, yeah.

Steve: Magnets are neat, yeah. And I got an error: SSL_ERROR_UNSUPPORTED_VERSION. And it said: "The page you're trying to view cannot be shown because the authenticity of the received data could not be verified. Please contact the website owners to inform them of this problem." And then it said, under Learn More: "This website might not support the TLS 1.2 protocol, which is the minimum version supported by Firefox. Enabling TLS 1.0 and TLS 1.1 might allow this connection to succeed."

Leo: So it's a really old server.

Steve: Yeah, exactly.

Leo: Using a deprecated protocol.

Steve: Well, I mean, things have been moving along, and Cool Magnet Man may be not very into his...

Leo: He's got other things, yeah, other things to do.

Steve: He's got other things, yeah. He's trying to figure out, wait, is this the North Pole or the South Pole? He's not sure. Anyway, so then it finishes, just like get ready: "TLS 1.0 and 1.1 will be permanently disabled in a future release."

Leo: But we knew that; right? I mean...

Steve: Yeah. Yeah, yeah, yeah.

Leo: I'm surprised that there's this escape valve. It's kind of in between; right?

Steve: Again, you know, we always see this. And we were sort of talking about this relative to the iframes last week, and sandboxing. As we're moving forward, we're often sometimes just leaving old stuff behind, or regretting some of the earlier decisions that were made. And you could argue that maybe the web wouldn't have happened unless things had been as completely crazy and wide open and anything goes as it was back then.

Leo: Well, it's not like TLS 1.0 or 1.1 was bad then; right?

Steve: No, they're just fine.

Leo: It's not broken. It's just weak.

Steve: Right, exactly. It's a little bit like the Spectre and Meltdown things where it's like, oh, you know, those pesky academics, they wrote a paper or two and said, oh, look, if you are facing the South Pole when you touch your nose and spin around three times, maybe you could decrypt this connection if you were lucky, and that kind of thing. So it's like, okay, again, makes sense to be moving forward. We're at 1.3 is the latest one. One of my servers actually is stuck at 1.2 because of course Microsoft doesn't want to - they want to force me to move to some Windows 10 server, which is like, ho ho ho, we'll see about that. Anyway, Mark Thompson tells me it's not so bad. Okay, well...

Leo: Is he trying to talk you down? Hey, I got this thing, Cool Magnet Man is...

Steve: Oh, god. I said to Mark, if they're done this to Windows 10, what have they done to Windows 10 Server?

Leo: By the way, CoolMagnetMan is still - see, the problem here, so there's kind of three layers. You could be supporting 1.3. You could be supporting 1.0, 1.1, and 1.2 and 1.3.

Steve: As I am, for example. GRC still supports the down versions.

Leo: A lot of sites do that. Right. And they don't want you to do that. But this is worse. He's not even supporting 1.2. He's not...

Steve: Correct.

Leo: Yeah. He's, like, really old.

Steve: Correct. It's interesting, too, because I was looking at this picture. I went over there and looked at his certificate. And his certificate says "not valid before August 4th of 2020." Which means that he did a cert renewal like last month, probably because that was happening. And as with all newly issued certs, it's only a one-year certificate. So anyway, I would imagine maybe he'll see his traffic fall off because some people are going to go, ooh, warning message, danger, danger. And it's like, okay, well, it's actually fine.

Leo: Should I not enable TLS 1.0 or 1.1? Is that risky to do? You know, they have that button. I could enable it.

Steve: Yeah. I pushed it because I went there, and I did not get the error message again.

Leo: I don't think he's updated this site in a while. Nice use of Microsoft WordArt.

Steve: Now, Leo, it looks very much like GRC. So be kind.

Leo: No, no, no. You don't have any WordArt on GRC.

Steve: Once upon a time. But no, yeah.

Leo: I'm surprised these letters don't rotate. I think Magnet Man's really missing a bet here. They could be going back and forth.

Steve: Oh, Leo, I'm sure that his Send Me Mail mailbox is spinning around somewhere.

Leo: Oh, I'm sure that's animated, yeah. No, we're not - we're just teasing you, Cool Magnet Man.

Steve: We're just showing our age is what we're doing.

Leo: Yeah, we recognize this site, yeah.

Steve: Yeah, it seems familiar. So Chrome, fortunately, will be getting tough on abusive ads. In a posting on GitHub, Google's engineer John Delaney has spelled out the Chromium Project's intentions regarding abusive ads. So first of all, modern web pages are a jungle of stuff. So how does Chromium, the Chromium engine, determine for itself what's an ad and what isn't? It comes down to something known as "ad tagging." Chromium is able to detect some ads and the resources they load in the browser. This enables the browser to measure the size, the performance, and the count of ads displayed to its users. It also allows the browser to intervene on the user's behalf when ads run counter to what they decide is the user's interest, for example, using a crazy amount of resources, engaging in some abusive behavior or whatever.

So the ad detection infrastructure they call "ad tagging." And it's not very inspired. It works by matching resource requests against a filter list to determine if they're ad requests. And in a sample that they've got of some code, they show them importing the EasyList, which of course is a well-known list that's being maintained by a community of known domain names that are providing ads. So they said: "Any requests matching the filter are tagged as ads. Further, requests, and some DOM elements such as iframes, made on behalf of previously tagged scripts, are also tagged as ads by the ad tracker." So it's not just images that match the filter, it's if scripts were coming from a known ad source, then things that are essentially descendants of those scripts would also be tagged as ads, which certainly you would want to have happen.

They said: "An iframe will be marked as an ad iframe if its URL matches the filter list, if tagged script is involved in the creation of the iframe, or if its parent frame is an ad iframe." So, you know, you can't sneak out of it by creating a frame within a frame and saying, oh, look, I'm not the original one. "The main frame on a page will never be tagged as an ad." Good. And then they said: "Any request made within an ad iframe is considered an ad resource request." So drilling down on this one level, we learn that the subresource filter loads the filter list and then performs this URL matching of any requests against that list. It's distributed, that is, the filter list is distributed via the component updater, which is just part of the Chrome installation. So it's being kept current constantly.

And the same list and component is also used for blocking ads on abusive sites and those that violate the better ads standard. They explain that each subresource request in the render process is processed by the subresource filter before the request is sent from the browser out. So it's not that it blocks things coming back. It never makes the request in the first place. It just denies it from the page making the request.

Okay. So you get ads identified as such. How are they treated differently? And this is where John explains what they call the "Heavy Ad Intervention. A small fraction of ads on the web use" - and John likes the word "egregious." We'll see this a couple times - "an egregious amount of system resources." He says: "These poorly performant ads, whether intentional or not, harm the user's browsing experience by making pages slow, draining the device's battery, and consuming mobile data," he says, "for those without unlimited plans."

And then he says: "In these egregious cases, the browser can unload the offending ads to protect the individual's device resources." He says: "This is a strong intervention that's meant to safeguard the user's resources with low risk because unloading an ad is unlikely to result in loss of functionality of the page's main content." He says: "Examples of observed ad behavior that are intended to be discouraged are" - no surprise - "ads that mine cryptocurrency; ads that load large, poorly compressed images," so just sloppy ads; "ads that load large video files before a user gesture; or ads that perform expensive operations in JavaScript, such as decoding video files or performing CPU timing attacks." Yeah, we don't want those.

So Google notes that it's not their intention to discourage any specific ad creative formats such as display video ads. So they're trying to be as agnostic as possible. So the user agent, the browser, will unload ads that use, and he says again, "an egregious amount of network bandwidth or CPU usage. We define 'egregious' as using more of a resource than 99.9% of ads as measured by the browser." So it sets a very high bar. And he says: "Only ads that have not been interacted with by the user will be unloaded."

And here's what's interesting, and this is some tech we've never talked about before that's therefore worth mentioning. All unloaded frames will be notified via an intervention report that the intervention occurred. This feedback is necessary to help advertisers or their ad technology vendors to identify and fix ads that are triggering this intervention. So first of all, just a little last word on the classification of ads. He says that's left to the discretion of the user agent. For example, Chrome detects ads using what we talked about, the ad tagging feature.

An advertisement is considered "heavy" if it has not been clicked on by the user and meets any of the following criteria: It uses the main thread for more than 60 seconds total; or used the main thread for more than 15 seconds in any 30-second window, so they said in parens "(50% utilization over 30 seconds)"; or used more than 4MB of network bandwidth to load resources. So any of those thresholds get crossed, the new Chrome technology will say nope and just boot the ad. Sorry, you're a bad ad. And he said that the thresholds above were inspired by the IAB's Lean Standard, that's in caps, but is chosen by looking at Chrome's metrics at the 99.9th percentile of network and CPU usage in ads.

So again, most ads are not going to cross that line. But those that do, and there are some, bye-bye. He said the threshold numbers were then rounded to be readable and memorable, thus 60 seconds instead of whatever it actually was, or 15 seconds in any

30-second window, and 4MB. Numbers were picked inclusive of all ads seen by Chrome's ad tagging, which is better at capturing display ads than native ads.

Okay. So when these interventions were put in place, they wrote that intervening at those thresholds saves 12.8% of the network usage by ad creatives and 16.1% of all CPU usage by ad creatives, with most of that value going to individual users, meaning individual sources. So that's interesting. That means that despite setting the bar that high, so that it is at the 99.9th percentile, it creates a savings of 12.8% in network usage and 16.1 percent across the board in CPU usage, which suggests that what is actually happening is a very small percentage of ads is trying to use a really obscene, I mean, I'll use that word rather than "egregious," an obscene amount of network and CPU. So this seems like a really good thing to do. You put the bar really high, and you say to a very small percentage of really extremely misbehaving ads, "Bad. We're kicking you out. You can't do this anymore."

So this one concept we had never talked about, the so-called "intervention report," is a very cool extension of the web API which essentially allows closing the loop. Modern web browsers support what's known as the "reporting API," which allows them to send back asynchronous event reports to a resources sourcing server, or at least the server that they have said they would like to have receive those reports. So as we know, a web page or code on a web page will make a query to a remote web server for content. And that content might be an advertisement.

One of the headers in that remote web server's reply can be "Report-To:." Its corresponding string value is a JSON formatted string which contains a URL and an "expires" tag, to which reports of any subsequent issues with that resource can be sent at any later time subject to the recipient's requested expiration. So, for example, if JavaScript were to crash which had been received from that source, a report can be sent back to the source saying, hey, your JavaScript crashed. Or in this case, if an ad has misbehaved such that it has been unloaded from the browser, Chromium-based browsers will, if there is a Report-To: header that accompanied the ad, will play nice and say, hey, thought you should know this ad has been booted. So you ought to consider maybe doing something different, like compressing the imagine or shortening the video or stop trying to mine cryptocurrency or whatever.

So anyway, just a very cool sort of next step in the evolution of the 'Net. One of the things that's nice about the way the 'Net works is because it is this relatively clean query and reply, or request-and-reply structure, it is simple to - carefully, they don't want to overdo it - but to carefully add additional high-value features which can further mature the web side of the Internet. And this is certainly one of those.

There is a fun way for our users to play with this stuff, for anyone who has Chrome 84 or later, which should be everybody because we're all at 85 now. There are a couple flags which are not yet enabled by default. Google will be, you know, this is like their statement of what we intend to do. But the tech already exists since Chrome 84. If you go to chrome: in the URL bar, chrome://flags/ and then #enable-heavy-ad-interventions, you can turn this on. And then also there's another one in there, #heavy-ad-privacy-mitigations, and that one is also hyphenated.

Setting the Chrome flags enable-heavy-ad-intervention will activate this behavior. But they recognized that because behavior could be probed by somebody deliberately trying to figure out where the triggers were, and that could also be a privacy problem, that the so-called "privacy mitigations" deliberately add some fuzz to the thresholds. So if you want to turn that off, you can turn that off.

Shoot, and there was another, I had another piece of this that didn't make it into the show notes because there's a site you can go to that deliberately misbehaves and so you can see this all happening, but I forgot to transcribe it into the show notes. So I will provide it next week in the errata. I meant to do that because it was a place you could turn this stuff on and then go there; and it was like, as I said, a deliberately misbehaving site that was kind of cool.

So, nice that our browsers are getting that. I thought it was really interesting that setting the bar that high created that large a savings for the typical user. So there are very few bad actors; but, boy, they are really bad. And now they're going to get caught and told, sorry, you don't get to do that anymore.

Malwarebytes Labs has issued a warning about their observation of a sudden large surge in attempted malware attacks leveraging oldie-but-goodie vulnerabilities in IE - yeah, IE - and Adobe Flash. These attacks are being observed on sites featuring, shall we say, highly explicit adult sexual content. One of the vulnerabilities from last year was CVE-2019-0752. Its description says this vulnerability allows remote attackers to execute arbitrary code on vulnerable installations of Microsoft Internet Explorer. User interaction is required to exploit this vulnerability, and that the target must visit a malicious page or open a malicious file.

Okay, well, visiting a page sets a low bar of user interaction. It's not like you have to do anything once you get there. The specific flaw exists within the handling of script commands that set certain properties of DOM (Document Object Model) objects. By performing actions in script, an attacker can trigger a type confusion condition. An attacker can leverage this vulnerability to execute code in the context of the current process.

Now, we talked about this at the time, a year ago - actually I think it was like a year and a half ago - and urged people to fix it. We talked about workarounds. There was one of those little 0patch quickie fixes. The point is that there are still IE instances that have not been updated. And for anyone unlucky enough to visit one of these infected websites with an unpatched version of IE, one of two well-known exploit kits will be downloaded and executed on that unfortunate visitor's machine. And if that one doesn't get you, the site will also try to leverage an even older flaw from 2018 that existed in Adobe's still not quite dead yet, you know, after all it's just a flesh wound, Flash Player.

Malwarebytes notified the industry about a malware malvertising campaign which is now in full swing. And the trouble is that a surprising number of people, hopefully none listening to this podcast, are still venturing out into the world, pushing a copy of IE ahead of them, and using that as their interface to the web. And as we know, it's just no longer up to the job. The group behind the attacks is a group named Malsmoke. They've operated on a scale far above other similar cybercrime operations and have, according to Malwarebytes, who's been tracking Malsmoke's attacks, abused practically all adult ad networks.

Most of the time, the group has managed to place malicious ads, as we call them, "malverts," only on mid-tier adult portals. But they recently "hit the jackpot," as Malwarebytes put it, when they managed to sneak malverts on xHamster, which - and this was news to me - is one of the biggest adult video portals in existence, and one of the biggest sites on the Internet.

Leo: I don't even want to ask what they post on xHamster.

Steve: Exactly. Billions of visitors each month, Leo, to xHamster.

Leo: For people who love rodents, I guess. Okay.

Steve: Ah, yeah. I would suggest no one go there, whatever that is, and certainly not with IE. But again, as I said, there's no way any of the listeners of this podcast are, like, choosing IE. You have to deliberately just want to go get hurt to use IE at this point. Although we also know, sadly, there are enterprise users who apparently still haven't moved. Or maybe at this point they are unable, for whatever reason, to move themselves away from IE because of vertical software which was written to it back in the day, and they're still having to use it.

There was also some news that flashed by me noting that Microsoft was going to begin pushing their Chromium-based Edge onto people's Win10 machines, whether they asked for it or not. At this point it's still been sort of an optional, hey, we've got a new Edge. It's better. Get it. So I would suggest that everyone should have asked for it long ago because it is better. There's just no downside unless some, again, misguided enterprise wrote custom code that only runs on the original Edge. And that one wasn't bad, but Microsoft has abandoned it. So everybody else needs to also. And in that case it's time to fix any software that an enterprise may have that only runs there.

So I really have no problem with having everyone moving over to a Chromium-based engine. That would be a good thing. And Edge has ported a new feature which they originally had in classic Edge over to the new Chromium-based Microsoft Edge. And that is there are some additional features available when you download a file. You need to enable it by going to edge://settings/downloads. You'll get a screen with almost nothing on it, which is refreshing, except an "Ask me what to do with each download," which you should turn on.

The reason is, as I had mentioned recently, with script being able to trigger downloads, you don't want those to be able to be dropped on your machine into the default download location without some sort of explicit involvement. Script can do it. But if you say, "Ask me what to do with each download," you'll get a big dialog saying where should it go? And just it's fine if you accept the default, click OK. But at least that way you're assured of being involved.

So anyway, the cool thing is that in the Chromium Edge, they're adding - and I had never noticed before, in the pop-up for options on the newly downloaded file there will be a "Copy download" link which could come in handy, and a "Delete," which is also nice. You don't have to go find it and then delete it if for some reason you downloaded it twice or three times or whatever. It's right there on the pop-up. It's like, no, don't need it after all, delete. So those are available, but only if you enable them at the moment with this "Ask me what to do with each download." That turns those additional context menu options on, which seems like a useful thing to me, especially because you want to be involved in downloads.

So we are the week after Patch Tuesday. It's the latest one that can happen because the month started on a Tuesday. And this was a big one. September, this month, saw 129 vulnerabilities patched. And some of those in the industry have admonished, try to be kind to your Windows admins, who will be scrambling to juggle the impact of last Tuesday's 23 critical vulnerabilities while working to resolve the bugs caused by patching them. This month's was not the all-time greatest patch month, but it did tie with this past June's 129 record-breaker. So now we have two months that are both - they now share the record of 129 vulnerabilities patched in a single month.

Microsoft provided patches to repair 23 critical flaws, as I said, in Windows and its related products; 105 important flaws; and one lone moderate vulnerability. While there were fortunately no zero-days that were known to be under exploitation, there were quite a few interesting vulnerabilities that could be remotely exploited for the enterprise, and of serious concern was CVE-2020-16875. That would likely grab the attention of any admin, being a memory corruption vulnerability in Exchange that can allow a remote attacker to perform remote code execution as the system account, meaning root, merely by sending a specially crafted email to an Exchange server.

So Dustin Childs, who's a researcher at Trend Micro's ZDI, their Zero-Day Initiative, wrote in his analysis last Tuesday: "That is about the worst-case scenario for Exchange servers." He said: "We've seen the previously patched Exchange bug CVE-2020-0688 used in the wild, and that requires authentication." This one doesn't. He says: "We'll likely see this one in the wild soon. This should be your top priority." And as we know, the danger, now that this has been patched, is that clever bad guys - and they are, I tip my hat to them, very clever - will examine the fix, then design and launch attacks before the Exchange admins can get their systems updated. So don't dally on this one, anybody who's in charge of Exchange server. It's been a week. Hopefully this is old news to everyone.

Another critical remote code execution vulnerability that should be prioritized for patching is 2020-1210. That one exists in SharePoint due to a failure to check an application package's source markup. It rates a 9.9 out of 10. I don't know, have we seen a 10? Anyway, it's as bad as any that we see on the severity scale. Satnam Narang, a staff research engineer at Tenable, wrote, he said: "To exploit this flaw, an attacker would need to be able to upload a SharePoint application package to a vulnerability SharePoint site. This vulnerability is reminiscent of a similar SharePoint remote execution flaw. That one was 2019-0604 that has been exploited in the wild by threat actors at least since April of 2019." In other words, bad guys really are getting on these, like immediately. It's a way in.

So this month there are a total of seven remote code execution bugs being fixed in SharePoint; and only one of them, 2020-1460, requires authentication. In other words, the other six don't. So yes, again, we see that patched vulnerabilities are irresistible to attackers who know that they will almost certainly find some laggards at high-value targets. So they're going to reverse engineer them. They're going to figure out how to exploit them. And then they're going to start looking for, in some cases Exchange, in this case SharePoint servers, and then have at it.

Justin Knapp, the product marketing manager at Automox, pointed to another critical remote-code execution vulnerability. This one has an 8.4, which was just fixed in Windows Graphic Device Interface, the GDI. That one's 2020-1285. And this one arises because of the way GDI handles objects in memory, providing both web-based and file-sharing attack scenarios that could introduce multiple vectors for an attacker to gain control of a system. In the web-based attack scenario, an attacker would merely need to craft a website designed to exploit the vulnerability, or of course post an ad somewhere on a heavily visited site. And that would cause the browser to render the ad. So users who viewed the website would get hit. Not a high bar on that one.

And we also have 2020-1129, another remote code execution flaw in Microsoft Windows Codecs Library, with an 8.8 severity score. As we know, state-of-the-art codecs are quite difficult to make perfect, yet perfect they must be. Any program that can cause Microsoft's HEVC codec to be invoked can be used to exploit this entry. An attacker could execute code on a victim machine by convincing someone to view a weaponized video clip. The flaw exists within the parsing of HEVC streams such that a crafted HEVC video file can trigger an overflow of a fixed-length, stack-based buffer. We've talked about this sort of problem in much detail in the past, so I won't go into it any further.

But, I mean, again, I said there were just lots of goodies - bad, bad problems that were fixed two weeks ago. There's also 2020-0922 which describes itself as a Microsoft COM for Windows Remote Code Execution Vulnerability that can be exploited by luring a visitor to a site with malicious JavaScript. So JavaScript running on a site could take advantage of it, if somebody figures out how to reverse engineer and weaponize that. So best not to find out. I'm sure by now all of our listeners have got their Windows 10 machines updated.

And I'll just finish with 2020-0908, a Windows Text Service Module Remote Code Execution Vulnerability that can be exploited by tricking a user to visit a site that contains malicious user-provided content or ads. Well, that's very generic sounding. People who looked at it were quite worried about its potential for exploitation. So oftentimes, again, they were not zero-days at the time of their release. Technically they can then never become zero-days, but they can certainly become one- or two- or three- or four-days. And users who haven't updated will be vulnerable to them until they do.

So, you know, it's tricky because we have been also, all year, we've been talking about updates causing problems for users who didn't have problems before. At the same time, they are now closing really bad and exploitable vulnerabilities. So I guess the best advice is, because they tend to be edge-case problems when they occur, you're probably better off updating; and, if something breaks, then quickly back yourself out of that so that you're not going to have that problem. So you tried to update if you could; but if something critical broke on your system, then you're just going to have to wait for Microsoft to fix that.

So speaking of eating away at things, remember that something broke in Windows some time ago that was causing it to forget that it had ever defragged its mass storage drives. So it was doing so needlessly and, in the case of SSDs, inducing unnecessary wear. Every time the system booted, it thought it had never defragged again. The idea is that it performs an initial defrag the first time Windows is rebooted after installation. And then I think it's, what, monthly it does it, but not every time. And it was also issuing trim commands, which only make any sense and are only supported by SSDs. It was sending trim commands to your spinning hard drives that were saying, uh, what? Trim what?

Leo: Oh, that's funny. Oh, that's hysterical.

Steve: Yeah, crazy. It's like, come on, Microsoft. Anyway, half of this was fixed last Tuesday, that is to say the excessive defragging of SSDs. Anyone who might have manually disabled the automatic maintenance of their SSDs in the interim to suppress this unnecessary wear - and I would think it's not that much wear, but still I know there are people who are like, oh, I don't want any writing that I don't have to have. If anyone did that, you can turn it back on because after last Tuesday's updates it now remembers that it had defragged it before. However, the mis-issued trim commands are still being sent to spinning hard drives. But it's not a critical problem. It simply fails and posts a note to the system's error log that for some reason your hard drive has rejected the trim command. It's like, uh-huh, yeah, it's supposed to because it doesn't know what to trim.

Last week we first covered this very bad problem in the very widely used WordPress File Manager. Remember that the problem was that files that were meant for development and were supposed to be renamed or have a name that would keep them from being active, had by mistake been left as a .php in several distributions. And as a consequence, many hundreds of thousands of installations of WordPress that had those versions of File Manager were trivially exploitable. And what we're learning is that it is the triviality of the exploit, it is how low is the hanging fruit that determines how rapidly and how widespread the script kiddies will jump onto this.

So I started off talking about it last week, saying it's not good when a zero-day flaw is discovered being actively exploited in an extremely popular plugin for WordPress. And it's also somewhat jarring that we keep covering exactly such news. So here we are again, revisiting it one week after last week's report, which is two weeks after the start of this drama because it began at the beginning of the month, on Tuesday, September 1st. Researchers at the WordPress security firm Defiant spotted more than 1.7 million WordPress sites being probed by bad guys, right off the bat, between September 1st and 3rd. That's what we reported last week.

In their updated report that was published as we were talking about it last week, last Thursday the 10th, their threat analyst, Ram Gall, wrote that the attackers have not stopped their siege. Anything but, in fact. Now the number of WordPress sites being targeted has jumped to 2.6 million. Multiple groups of bad guys are known to be targeting this File Manager vulnerability, though Defiant has noted that there are a particular two from among the many who have seen the most success in deploying their malware. And one reason is they're taking the time to close the backdoor to the site after they've entered.

One of the two is an attacker known as Bajatax. He's based out of Morocco. He's known for stealing user credentials from specific ecommerce websites. And in this instance, once the attacker compromises a WordPress site, he injects malicious code that harvests and exfiltrates user credentials via Telegram on any subsequent normal front door login to the site. And then those credentials are sold to the highest bidder on the dark web.

The second attacker has been observed to inject a pair of backdoors onto compromised WordPress sites, compromised by having this vulnerable version of File Manager. One of those backdoors is put into a randomized folder name and another onto the site's web root. Both are camouflaged as .ico, to look like icon files, to reduce the likelihood that the site's admin will find either or both of them and thus curtail the attacker's subsequent access to the site. And as we've long said, once a server, a site, or a machine has been compromised, it can never fully be trusted again.

The PHP infector that's being used by the second attacker is a variant of an infection that's been previously used to deploy cryptominers and to run SEO spam campaigns against compromised sites. And Defiant, the WordPress security firm, has observed both of these attackers working to block, as I mentioned, other attackers' exploit attempts by password-protecting that exploitable, it's connector.minimal.php file - actually, I think it's connector-minimal.php - on the sites once they have infected those. Defiant's people said that their site cleaning team had cleaned a number of sites compromised by this File Manager vulnerability, and in many cases discovered malware placed there by multiple attackers, obviously by attackers who did not think to close the door behind them after they had entered. But thanks to those first two being proactive about blocking others and collectively employing several thousand IP addresses in their attacks, those two have been the most successful.

And somewhat amazingly, overall, Defiant's researchers have monitored attacks attempting to exploit this vulnerability originating from more than 370,000 separate IP addresses. So they're exploiting proxies that they've been able to compromise, or maybe botnets, which are deployed and are ready and able to launch attacks under control. But attacks are coming from everywhere. And at this point, two weeks downstream, it's probably safe to say any WordPress site that was using a vulnerable version of this File Manager add-in that didn't immediately update it now has malware running on it. And maybe it's peacefully coexisting. We don't know what the malicious guys are doing.

We know that in one case the valid login credentials are being harvested. So as soon as the legitimate owner of the site logs in, if you have been compromised by the first of those two, the Bajatax guy, then your valid login credentials and the site you're logging into are sent off via Telegram. So they're encrypted and received by the attacker, who then puts them up for sale on the dark web for anybody else who wants to log in as the valid user.

So again, if you find that your site is compromised, you just sort of have to flush it. Unfortunately, maybe you back up your content. You just destroy the site. You don't reuse your previous credentials, obviously. And then build a new site from scratch and then reload the content, that is, the textual content, not files that have been - don't just restore the whole site or you could be restoring the malware, too. It's a mess.

Zoom, that we've been talking about ever since it became the talk of the town thanks to COVID-19 and the coronavirus, now is offering two-factor authentication, which I think is just all for the best. Last Thursday, the 10th, Zoom announced the availability of several forms of second-factor authentication - traditional time-based token, standard two-factor authentication, or via SMS, or a phone call for logging into Zoom.

I have a note, I'm sorry, a link to all of the details, but I'll just quickly explain that they said log into the Zoom dashboard. You navigate to the Advanced, then the Security feature. And then you will find a new sign-in with two-factor authentication option. You enable that, and then you've got three options for how pervasive to require two-factor authentication. You can choose to require it for all users of the account; for only users with specified roles, and if you choose that option then you choose which roles will be required to use two-factor authentication; or you can enable it for users that are in specific groups and then choose which groups you want to have to require two-factor authentication. And then as always with a web form, save the changes that you've made to make them permanent.

So anyway, I think that's just a nice move forward for Zoom, which is continuing to be very popular. And to prevent the problem of an easy username and password hack, as we know, a second factor makes sense. People could still not understand how important it is to have really strong passwords, be using a simple password, and then be subject to a brute-force login attempt.

Okay. The one that didn't make the cut for being one of the title interventions is Raccoon. One headline read "New Raccoon Attack Could Let Attackers Break SSL/TLS Encryption." But a more sober headline in the tech press covering this said "Raccoon attack allows hackers to break TLS encryption under certain conditions." And then their subhead further noted: "The Raccoon attack is described as 'really hard to exploit' and its conditions as 'rare.'"

So anyway, as I said, when I was gathering stuff together, I initially thought I would include this in the headline, but no. What the researchers found was an attack which no longer works in TLS 1.3, for one thing. So it's 1.2 or earlier. It is a very subtle edge-case timing-related attack in the instances that ephemeral Diffie-Hellman key exchange is used to set up the premaster secret in a TLS connection. And for anyone for whom those terms are not known, we have done a number of podcasts in the past where we talked all about how secure connections are set up. At the time it was probably over TLS 2.0, so it's a ways back. But all of the terminology and techniques are the same.

So these guys wrote: "Diffie-Hellman key exchange is a widely adopted method for exchanging cryptographic key material in real-world protocols such as TLS-DHE. Past attacks on TLS-DHE focused on weak parameter choices or missing parameter validation." So that would be implementation scale sorts of problems. He said: "The confidentiality of the computed Diffie-Hellman share, the premaster secret, was never questioned. DHKE (Diffie-Hellman Key Agreement) is used as a generic method to avoid the security pitfalls inherent in TLS-RSA."

They said: "We show that, due to a subtle issue in the key derivation of all TLS-DHE cipher suites in versions up to [and including]" - I added the "including" because they didn't make that clear - "TLS 1.2, the premaster secret of a TLS-DHE session may, under certain circumstances, be leaked" - that's not good - "to an adversary. Our main result is a novel side-channel attack, named Raccoon attack, which exploits a timing vulnerability in TLS-DHE, leaking the most significant bits of the shared Diffie-Hellman secret."

Anyway, I'm going to skip the rest of their discussion because it doesn't matter. Which is why it's also not in the title of this podcast. It requires an obscure modulus to be used for the Diffie-Hellman key agreement handshake. It also requires that the attacker be immediately adjacent to one of the endpoints in order to have any chance of having sufficiently accurate timing on the handshakes. It is a side-channel attack because it involves timing.

Even the researchers were quoted saying: "The vulnerability is really hard to exploit" - that's their words - "and relies on very precise timing measurements and on a specific server configuration to be exploitable. The attacker needs to be close to the target server" - this is them still speaking - "to perform high-precision timing measurements. The victim connection needs to use ephemeral Diffie-Hellman, and the server also needs to reuse ephemeral keys. And, finally, the attacker needs to observe the original connection."

In other words, it's useful research, but it's nothing for us to worry about in the real world. If you see headlines or mentions of this, just flip to the next one because, fine. But all that said, we know that attacks only ever get better, as Bruce Schneier famously said. Maybe this could. And we're going to be talking about exactly things getting harder or better when we get to the title meat of this podcast. So Microsoft, Mozilla, OpenSSL, and F5 Networks have all recently released security updates to block Raccoon attacks.

So as I said, it made a change. Things are better now. Useful academic research. But it would have been very difficult at this stage of its development for this to actually affect anybody. And while I'm on the subject, by the way, don't google "raccoon attack" for additional information. Turns out those little guys are much scarier than the actual digital attack that bears their name. I did google "raccoon attack," and whoo.

Leo: Did you find Kevin Rose? Did you see his raccoon battle?

Steve: No.

Leo: Oh, I'm surprised. Google "Kevin Rose raccoon attack."

Steve: Uh-oh.

Leo: He and his dog, Toaster, got in a bit of a fracas some years ago. Here, I'll just play the 47-second video of it from ABC News.

Steve: Wow.

Leo: This was some years ago. "Man saves dog from raccoon attack: At 1:00 a.m. I heard my dog Toaster crying and yelping in pain. I discovered a raccoon attacking him. I do not encourage" - Kevin is a very sweet, gentle person. So he says: "I do not encourage animal violence. I wanted to get the wild animal as far away as possible. Toaster is okay." So there's Toaster and the raccoon battling it out. And here comes Kevin Rose, superhero. He picks up the raccoon by the tail and throws it. Toaster escapes unscathed. Kevin chases the raccoon off into the night.

Steve: Yeah, he really did fling it.

Leo: You want another shot?

Steve: I'm glad the raccoon was able to continue moving.

Leo: Apparently he says the raccoon - this is so funny. He says the raccoon wandered off unscathed after it. But I'm not surprised. Here's another angle. Whoa.

Steve: Whoa. What a little adrenalin will do for you.

Leo: We've been mocking Kevin ever since. That was 2015, I think, five years ago. 2013.

Steve: I'm glad he saved Toaster.

Leo: He saved Toaster.

Steve: Glad he saved Toaster.

Leo: The only dog I know that has an Instagram filter named after him.

Steve: One last bit, a quick update on SpinRite. I am proud to say that I'm down to very few, but not quite zero, remaining edge cases with the mass storage benchmark that are occurring on a few very specific pieces of hardware. I briefly put that journey on hold because I wanted to actually finish getting GRC's new web forums ready to go so they would be ready as soon as the code was. So I spent some time over the weekend working to finish bringing them up. The last thing that needed finalizing is to get SQRL logon working for that domain. Turns out it was a little tricky.

When I wrote GRC's server-side support for SQRL, I made it flexible enough to handle any one domain. But I did not build in support for multiple simultaneous domains. Sqrl.grc.com and forums.grc.com are separate authentication domains as far as SQRL is concerned because anyone using SQRL is asked to confirm the domain they're logging into, so it should appear as sqrl.grc.com for the SQRL forums and forums.grc.com for the GRC forums. So anyway, I added support to GRC's server-side implementation to handle any number of explicit subdomains of a parent domain, so we have that capability now.

And then something still wasn't quite right, so Rasmus - remember Rasmus Vind is the terrific web developer who knows the XenForo system and PHP upside down and backwards. He and I spent a bit of time yesterday using Signal to communicate, as we had a couple years ago when I was first working with him, to figure out what might need to be updated. I had updated to the latest version of XenForo, and many plugins needed to be tweaked a little bit after that. So his may need to be revisited, but I'm sure we'll have it working within another day or so. So anyway, at that point I'll be getting back to the benchmark, get it finalized, get it packaged, and then be able to say to all of our listeners on this podcast, I've got something really cool for everybody to play with.

Leo: Awesome. Can't wait. All right. Let's talk about BLURtooth, blue whatever that is.

Steve: Okay. So BlindSide first. And these guys are going to be familiar to us. A team consisting of five super sharp researchers at ETH Zurich, the Stevens Institute of Technology and the University of Amsterdam, including our old friend Professor Herbert Bos, has just managed to oh-so-cleverly leverage Spectre-style processor performance optimizations in such a way that, first, they bypass any attempts to mitigate the processor's Spectre-style vulnerabilities; and then, second, leverage those vulnerabilities to successfully perform a so-called BROP (B-R-O-P), which stands for Blind Return-Oriented Programming. And this can be done in the face of the best KASLR (Kernel Address Space Layout Randomization) while completely suppressing any "wrong layout guess" crashes.

Okay, so let's back up a bit. I'll explain what they've done because it's - when you see the video, it is this week's Security Now! video, a four-minute video showing their proof of concept, which they start on a Linux terminal. And in four minutes it has root. So now I have your attention.

Leo: That's not good.

Steve: No. Okay. So let's back up a bit. You'll see, in fact, you have the video onscreen. They first do a "who am I," and they get some random username. We'll see four minutes later, after their system is finished cranking, they do another "who am I," and you know who is the...

Leo: Somebody else, huh.

Steve: Then logged in, yeah. Okay. So a previous work by researchers at Stanford University was titled "Hacking Blind." And I've got a link to the abstract, or to their whole paper, but the abstract describes what "hacking blind" is. They said: "We show that it is possible to write remote stack buffer overflow exploits, without possessing a copy of the target binary or source code, against services that restart after a crash. This makes it possible to hack proprietary closed binary services, or open source servers manually compiled and installed from source where the binary remains unknown to the attacker."

That's something that we've never really touched on before is that the return-oriented programming assumes that you absolutely know the system that you're attacking, meaning you know what the ends of the subroutines that you're attempting to leverage with your so-called "gadgets," you know what they are. But that's only the case when you know what the binary is. If you don't have an exact binary, you're stabbing in the dark. And that's exactly the point. That's what they meant by "hacking blind."

So they said: "Traditional techniques are usually paired against a particular binary and distribution where the hacker knows the location of useful gadgets for Return-Oriented Programming. Our Blind ROP (BROP) attack instead remotely finds enough ROP gadgets to perform a write system call and transfers the vulnerable binary over the network, after which an exploit can be completed using known techniques." In other words, they're able to, by tolerating the fact that they're going to, like over and over and over, be crashing the service, because the service restarts - well, in fact I'm stepping on this.

They said: "This is accomplished by leaking a single bit of information based on whether a process crashed or not when given a particular input string. BROP requires a stack vulnerability and a service that restarts after a crash. The attack works against modern 64-bit Linux with address space layout randomization (ASLR), no-execute page protection (NX) and stack canaries." So a fully instrumented state-of-the-art attack. And Leo, I see that it is just about to finish.

Leo: Who am I?

Steve: Here we go.

Leo: Who am I? I am Root. Ha ha ha. Or Groot.

Steve: Oh, baby.

Leo: Wow. So they got root.

Steve: They got root.

Leo: But that's really interesting because they don't know the binary, but they are able to do it by probing?

Steve: Okay. So that first work that I mentioned, the work that Stanford did, it required a service that would auto restart.

Leo: Right.

Steve: So they were crashing it over and over and over.

Leo: Oh, I get it. I get it.

Steve: And each time, the one bit of information they obtained was did it crash or not.

Leo: Right.

Steve: And so, okay. So on the other hand, as we've often noted, when a system contains a vulnerability, attempts to exploit that vulnerability more often than not simply crash. And in fact that's what fuzzers do; right? They discover potentially weaponizable flaws by first crashing a system that should not be crashable. Then humans examine the execution path that the fuzzing triggered to see whether it might be exploitable. And as we know, one of the keys to spotting when a system might be under active attack is when the system's logs suddenly show that normally reliable processes are suddenly crashing and auto restarting for no obvious reason. Yeah, it could be buggy code.

Leo: Somebody's doing something.

Steve: Exactly. But why would buggy code suddenly choose to start crashing? It could be the result of someone attempting and repeatedly failing to guess at the location of a known vulnerable module in an OS kernel that employs KASLR, whose kernel modules are randomized to occupy a different location in RAM for each boot. And that's of course done to specifically thwart this sort of hit-or-miss guessing attack. You would rather have the system crash than let the guy gain root on your system.

Okay. So with that background, what this team, the BlindSide guys have done, this team of five brilliant researchers, Herbert Bos among them, the guys that nailed the Rowhammer attacks years ago, what they've managed to do is to figure out a way of completely neutering that powerful and arguably vital protection provided by KASLR by leveraging - get this, Leo - a system's Intel processor Spectre optimizations to repeatedly probe for the location of a known kernel bug in memory without ever crashing the system.

Leo: Uh-oh.

Steve: Uh-huh.

Leo: Uh-oh.

Steve: That's what you just watched in that video.

Leo: There was no crash.

Steve: You watched a zero crash probe for a known bug that they were then able to leverage to give themselves root, and nothing crashed.

Leo: That's not good.

Steve: So their paper is titled "Speculative Probing: Hacking Blind in the Spectre Era." And the video, for those who don't know, grc.sc/784, the number of today's podcast, 784. Https://grc.sc, for shortcut, /784. That will bounce you to a YouTube video where you can see this happen for yourself. And we've often quoted Bruce Schneier saying "Attacks never get weaker, they only get stronger." Here, after about 21 months of awareness and exploration of Spectre, and what Spectre can do, they are now leveraging Spectre to perpetrate an extremely practical attack. So, wow, yeah.

Leo: How would they get it on the server, though? That's the question.

Steve: That is true. The Stanford attack was a remote attack that would cause crashes and then export the binary, which would then allow them to design a specific attack. Here you would need to have code running on the server. On the other hand, if it was a VM, the problem with Spectre is that it's a cross-process leakage. So you could, for example, probe the hypervisor in order to penetrate its protections in order to gain that advantage. So anyway, it just demonstrates that Spectre is no longer just like a lab curiosity for the academics. Oh, and I forgot to mention the source for that is on GitHub.

Leo: Oh, wow. Okay. Well, get to work.

Steve: Buckle up.

Leo: Geez, Louise.

Steve: Yeah. Uh-huh.

Leo: Okay. Well, I'm just going to put a padlock on my server, that's all there is to it, right there.

Steve: Secondly, we have completely different BLURtooth. Here, they have four headlines: "BLURtooth Vulnerability Lets Attackers Defeat Bluetooth Encryption"; "Bluetooth Bug Opens Devices to Man-in-the-Middle Attacks"; "BLURtooth Vulnerability Lets Attackers Overwrite Bluetooth Authentication Keys"; and, finally, "New Unpatched Bluetooth Flaw Lets Attackers Easily Target Nearby Devices." Everybody gets the idea that we've got a problem with Bluetooth.

BLURtooth is the result of insufficiently strict requirements appearing in previous Bluetooth specifications, from 4.0 through 5.0. Some tightening recommendations appeared in the minor 5.1 update, which was released in January of 2019, so a year and three quarters ago. The problem is that it offered some weird additional features like direction finding for Bluetooth, which, you know, if you didn't need that, you just ignored it. So 5.1 said, you know, we've decided some things should be tightened. And by the way, here's a bunch of cool new tech. But a lot of people just said, okay, we don't need that, and they stayed with 5.0.

So technically the mistake was in allowing for cross-transport keying of pairing associations. And I'll explain what that means, of course. Devices that support two different transports, specifically both Bluetooth Low Energy (BLE) and also the Basic Rate/Enhanced Data Rate, the so-called BR/EDR transport methods. Devices that support both, that is, BLE and BR/EDR, are known as "dual-mode" devices. And the earlier specs, as I said, allowed for those two differing transports to deliberately affect each other's pairing keys. This was designed deliberately because in fact it even - that protocol even has its own name and abbreviation. It's called Cross-Transport Key Derivation (CTKD).

It turns out, as originally implemented, and as presently implemented by most dual-mode devices, it's not as secure as they thought it was. CTKD pairing allows the devices to pair once using either transport method while generating both the BR/EDR and the BLE long-term keys, without needing to pair a second time. So back when the committee was working on the Bluetooth protocols, they said, you know, why make them pair twice? If the devices are dual-mode, and they're able to support both BLE and BR/EDR, let's just pair once. And then we'll use this CTKD, this cross-transport keying mechanism, to pair the other side, the other mode.

So once again we get bitten by the desire for convenience because dual-mode devices using CTKD are able to overwrite the original long-term key or the link key in cases where that transport was enforcing a higher level of security. Okay. So for example, the Bluetooth spec introduced so-called "Just Works" pairing, right, for those instances where easy communications but less security is all that's needed. So, for example, fitness trackers; maybe Bluetooth LE-enabled jewelry; devices used to track a pet tag; or other technologies where you're just not concerned with sensitive information, like a credit card or health data. You're not dealing with content. You're just dealing with, like, oh, you know, how far away has Rover or Toaster gone? Or is Toaster being attacked by a chipmunk, or something worse, a raccoon?

Anyway, so this Just Works technology is part of Bluetooth LE. Even when devices also support the deliberately far more secure BR/EDR pairing. As we know, one of the security enforcements for pairing is requiring a time-limited human-triggered event or some out-of-band communication like where you use a PIN, which is not part of the wireless protocol, to enable high-security pairing. But two separate security research groups both independently discovered that it's possible to leverage that simple, unsupervised, nonsecure offering, the so-called "Just Works" pairing, and use then the CTKD, the cross-transport key derivation to break into and insecurely rekey the secure transport side. And that was never intended.

So that's a big whoopsie. There's no doubt that there are places where this would be a problem. The implementers of Bluetooth-connected systems might have deliberately implemented both transports under the entirely reasonable assumption, I mean, and it is so stated in the spec, that the two transports were cryptographically isolated from one another, as indeed they were intended to be. So, for example, imagine that some industrial control system uses a one-time highly secure pairing to communicate with its controlling monitor system. But as a convenience, it also allows for Just Works pairing to be used with any nearby smartphone for read-only passive monitoring. That's an entirely reasonable architectural design decision.

But now, unfortunately, any of those nearby unauthorized smart phones which are able to pair over Just Works Bluetooth Low Energy, now have the means to intercept, rekey, and perform an active man-in-the-middle attack against the main control link that was assumed to be super secure. So for those using Bluetooth features of versions 4.0 through 5.0, as I mentioned before, there was no push or rush to implement the tighter restrictions which were added in 5.1. If 5.1's additional functional enhancements were not needed, then why bother, since 5.0 was assumed to be secure?

That all changed last week. So we can expect to see a move to v5.1 on any dual-mode Bluetooth devices, you know, like all of our smartphones that also now offer BLE. So as always the case, those Bluetooth devices that are not part of some actively maintained upgrade cycle like, gulp, maybe some process monitoring hardware which is no longer being maintained or dynamically updated, everything that is stopped at 5.0 will forever remain vulnerable to this newly revealed class of cross-transport attack.

And so what we are seeing is we are gradually creating a growing environment of long-term vulnerabilities that are more than likely never going to get fixed. And you just have to think that there are state-level actors and agencies that are not missing a single one of these. They are adding to a large and growing portfolio of ways to attack this, ways to attack that. And these things are never going to get fixed. So a bit of a brave new world that we are creating really as a consequence of the unfixable debris that we are leaving behind as we're continuing to better learn how to do these things as we move forward. Unfortunately, we're depending upon the ability to fix our mistakes in the past, and that's a strictly time-limited ability.

Leo: Well, as always, you've cheered me up immensely, yes. BLURtooth and BlindSide.

Steve: Yup. Well-named.

Leo: That's Steve Gibson. Yeah, yeah. It makes for a good show title, anyway. Steve Gibson is at GRC.com. That's his website. That's where you can find, of course, copies of this show, including 16Kb versions for the bandwidth-impaired, and full transcriptions so you can read along as you listen. GRC.com. While you're there, check out SpinRite, the world's finest hard drive maintenance and recovery utility, supports SSDs too, soon will support even more. He's working on 6.1, and if you buy now you'll get a free copy of 6.1. So this is a good time to pick up your copy of SpinRite at GRC.com.

He's @SGgrc on the Twitter, if you want to leave him comments. Lisa and I both get email for you all the time. And I just say, you know, every end of every show we tell you, you can tweet him, @SGgrc. Or really old school, go to GRC.com/feedback, leave him a feedback form. That's the easiest way to do it. I don't know how to get a hold of Steve. He lives in the Fortress of Solitude somewhere in Orange County. I can't...

Steve: Getting work done, yup.

Leo: He doesn't want to hear from anybody, I can guarantee you that. He's got stuff to do, places to go, people to see, code to write. Assembly code, at that. You will find the show also at our website, TWiT.tv/sn. There's a YouTube channel devoted to Security Now!. I don't know what the exact URL is, but if you go to YouTube.com/twit we've got all the links there in the sidebar. And of course the best thing to do is subscribe. That way you'll get the podcast the minute it's available, and you could begin your collection of all 784. Make sure you get the very rare upside-down printed #83 edition. That's something. Worth many millions of dollars.

We do the show every Tuesday, right after MacBreak Weekly, more or less 1:30 Pacific, 4:30 Eastern, 20:30 UTC. You can watch us work live, unlike your local mailman. We have a live feed of our efforts at TWiT.tv/live. If you're watching that live stream or listening, there's a number of places you can push the button and get it. You can also be in the chatroom at irc.twit.tv. They're watching and listening live, too: irc.twit.tv. Steve, have a great safe week, and I'll see you next week right here on Security Now!.

Steve: Okay, buddy. Bye.