Zippity Do or Don't

Description: This week we update again on VPNFilter, look at another new emerging threat, check in on Drupalgeddon2, examine a very troubling remote Android vulnerability under active wormable exploitation, and take stock of Cisco's multiple firmware backdoors. We discuss a new crypto mining strategy, the evolution of Russian state-sponsored cybercrime, a genealogy service that lost its user database, ongoing Russian censorship, and another Adobe Flash mess. We check in on how Marcus Hutchins is doing. And, finally, we look at yet another huge mess resulting from insecure interpreters.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-667.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-667-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We have a lot to talk about. VPNFilter turning out to be worse than we thought and affecting more routers than we thought. And then there's a flaw in a protocol used by dozens of programs. You probably use it every day. The problem with zip files, next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 667, recorded Tuesday, June 12th, 2018: Zippity Do or Don't.

It's time for Security Now!, the show where we get together with Mr. Steve Gibson, right here over my left shoulder, and talk about security, privacy, technology. Hi, Steve.

Steve Gibson: Hello, Leo. Great to be with you again as we plow into the final third of the podcast. We're on 667.

Leo: It's only the final third if you know that we're only going to 999. Now, somebody did point out that you could always use hexadecimal and stay in the three-digit range.

Steve: Yes. I've had many helpful suggestions. Others have suggested, well, we never did zero, and we haven't gone negative yet. And I try not to go negative.

Leo: No, no.

Steve: I just think that that's really - that doesn't really serve our listeners. And, after all, that's why we're here. And with a title such as "Zippity Do or Don't"?

Leo: What's that all about?

Steve: Yeah. Well, it turns out that, as one of our recurring memes here, we note how difficult it is to get interpreters right. And it turns out that unzipping, that is, decompressing, is a widespread, very, well, important and crucial process to get right. And if you make a mistake in your decompressor, it can have quite widespread ramifications. And in fact that has been found. So we will wrap up this week's podcast with Zippity Do or Don't. In the meantime we're going to get an update on, unfortunately, VPNFilter, whose - I hope everyone's sitting down for this - grasp has expanded from 16 identified make and models of routers to I think the number was 71. So, yeah.

Leo: Oh, wow. I knew that was - I was not sanguine that the list we had was complete.

Steve: Yeah. And in fact I think at this point we have to assume that not being on the list no longer confers any proper sense of safety. So we've got an update on VPNFilter. We've got another about 40,000-strong network threat that is emerging. We need to check in on Drupalgeddon2, now that it's been several months and things have probably reached steady state, to see where it is. We have a very troubling remote Android vulnerability under active wormable exploitation. And in fact that is that chart. Our Picture of the Week is...

Leo: I don't know how you didn't make that the title of the show because "Active Wormable Exploitation" is an excellent title. It's a really felicitous phrase.

Steve: Well, and the acronym is AWE, A-W-E, so it's like, ooh, we're in awe now, baby. Also we need to take stock of Cisco's recent problems with multiple backdoors. Not just one, not just two, not just three, but four that we know of. And it's like, okay, what has been going on over there in Cisco land that somebody has been deliberately putting backdoors in Cisco products?

Leo: Huh.

Steve: Yeah. Also we're going to take a look at a new crypto mining strategy. The evolution of Russian state-sponsored cybercrime. Those who watch have been noticing that Fancy Bear has been dancing a little differently recently. A genealogy service that lost its user database. But why at the same time that sort of upsets people, it's like, oh, my god, my DNA, but doesn't have anything to do with DNA. And this outfit, MyHeritage, did everything right. Also a look at Russia's continuing march toward censorship, fighting

the technology that the Internet was supposed to give us. Another look, or a look at another Adobe Flash mess that caused them to release an emergency update last Thursday.

Also a check-in on how our buddy Marcus Hutchins is doing. Things were looking better, and then the government slapped him again. And I just wanted to sort of touch base since, as we know, he was nabbed at McCarran, the Las Vegas airport, as he was trying to go home after last summer's hacker conferences. And, finally, another huge mess we will wrap up with, resulting from insecure interpretation of data structures and how that could be manipulated, thus Zippity Do or Don't.

Leo: You know, ages ago I wrote, before there was Zip, before there was StuffIt, before there was...

Steve: Remember Arc. Arc was the early one.

Leo: Arc, okay. So there was Arc for Windows, but there was no Arc for Mac. So I wrote, and probably you'll be proud to know in assembly code, I'm pretty sure it was in assembly code, an Arc for Mac. But it wasn't a compressor, it was just a decompressor. And I'm trying to remember, but I think it was kind of a table lookup process that was involved.

Steve: Probably would have been Huffman code, and that would be table lookup.

Leo: Yeah, yeah. It was probably Huffman as opposed to Lempel-Ziv. I don't know if Lempel-Ziv was out yet. That's when everything changed, when Zip came along, is Lempel-Ziv. But yeah, it was probably Huffman run-length encoding. And so a table would be the fastest way to do it. And I can't imagine any way that that could be risky, but I'm sure you'll - I don't have my source code to check. But I'm sure you'll come up with some way.

Steve: Well, Leo, if you had your source code, we'd have to find something that it ran on. That, of course...

Leo: Yeah. You got a 68000 lying around?

Steve: That's the problem. Beautiful chip.

Leo: It was a beautiful chip. It had a flat memory model. I didn't have to do any of that weird stuff you have to do on your x86 platform.

Steve: And an orthogonal instruction set.

Leo: Orthogonal, yeah, whatever that means.

Steve: Everything applied to everything, whereas Intel was just an abomination.

Leo: Oh, the 68000 was a beautiful chip to code assembly language for.

Steve: I really wish it had won the war. There was a battle there for a while, and it gave out.

Leo: Yeah. It was my - I had done a little 8086, and I hated it, mostly because of memory segmentation. And I just, when I got the 68000, it was like, oh, this is beautiful. It was actually fun to write assembly code for that. And Apple had a really good tool, the Macintosh Programmer's Workshop, that did a lot - it was like MASM. It had a lot of macro capability. So you really were working almost in a high-level language. Once you wrote your macros, it was pretty easy to work with. Otherwise I would never have been able to do it.

Steve: So not pleasing us is the march forward of the VPNFilter malware.

Leo: Wow.

Steve: All evidence is that this is Russian state-sponsored cyberwar, cybercrime. The fact that there's a strong bias toward routers and IoT devices in Ukraine is what leads a lot of people, and there's been other breadcrumbs that security researchers have also tracked back to known IP addresses and command-and-control servers and so forth that are known to be controlled and have been used in the past. So Russians, the group that are doing this, that APT28, a.k.a. Fancy Bear, and it's known by a whole bunch of different names, they're not even really trying to keep themselves clean in terms of no accountability for this. And of course the Russian government denies any knowledge. Oh, this is fake news. It's like, oh, wonderful.

So what we have is the Talos threat intelligence group of Cisco, who were the original prime discoverers and trackers, have continued watching this and have further reverse-assembled the code. Remember that, while there are oftentimes hints left behind in code, it's very often just ones and zeroes that are running on a given device. So you need to know what device it's running on. You need to have the ability to disassemble the binary into the assembly language for the device. Sometimes it's interpreted code, which can make that task easier. But the point is just saying oh, look, there's malware in this router, it's like, okay, what does it do? Well, it's mal. But to know exactly what it does takes time. And so it's not at all surprising that several months have gone by, during which I'm sure that hackers, white hat hackers inside of the Talos group have continued to study what it is that they've found.

So as they've been watching it, the first thing that they have found is that, to the list, the previous worrisome but kind of modest list of Linksys, MikroTik, Netgear, TP-Link and QNAP NAS devices, we must now add routers by ASUS, D-Link, Huawei, Ubiquiti, UPVEL, and ZTE. So essentially we've gone from, and I have here in my notes, I did quote them

earlier from memory correctly. We've gone from 16 makes and model numbers to at least 71, 71 we know of.

And in fact it just didn't make sense for me to list them in the notes here. Normally I do, but the list is too long. So what I have in the notes is a link to the talosintelligence.com blog. I imagine you could just go to blog.talosintelligence.com, and you would find vpnfilter-update is their URL there, if you were curious. But as we were saying at the top of the show, Leo, not being on that list really should not confer any great sense of security any longer.

Leo: I was worried about that even from day one, that that list wasn't complete.

Steve: Yeah, I mean, this is a determined group.

Leo: Do you think that it is that they are similar chipsets, similar firmware? Or that they're extending its capabilities? And furthermore, didn't the FBI discontinue, cut off the servers, the routers?

Steve: Yeah. And you gave me a perfect segue because what I had here in my notes to remind myself, I wrote, "Also frightening, due to the R&D and development resources it implies, is the presence of many device-specific modules."

Leo: Oh, boy.

Steve: This is not a one-size-fits-all opportunistic shotgun. This is a, quote, "We want," literally, "We want to perform passive packet sniffing of all traffic passing through a TP-Link R600-VPN router." And the Talos group found exactly such a module.

Leo: Would that crack, if you're using it with a VPN, would that crack into your VPN? I guess it would.

Steve: Yes, yes.

Leo: And expose your traffic, wow.

Steve: Yes. What they found was, in their additional watching of I'm sure their honeypots and tracking down the network, remember that this thing has three stages. Stage one is the little sort of the hub, the little kernel, that plants itself in the firmware. And they made it small to increase the probability of it being able to become persistent. The larger it is, the less chance there would be a space in the file system for it to live. So they kept it very lean and very small. And then it reaches out to, it turns out, a much larger number of locations than was believed, which is one of the things that has allowed it to stay alive.

Leo: So not ToKnowAll and Photobucket.

Steve: Yeah. There were many more locations.

Leo: I thought that was a little suspicious. That was a little too easy.

Steve: Yes. And it does also open a listener, which allows the bad guys to access it directly.

Leo: You mentioned that, that that's why you have to rewrite the firmware.

Steve: Right. So they had been monitoring their network, building up a list of IP addresses. And when the FBI did its takedown, they said, oh, okay, and they just...

Leo: Big deal.

Steve: ...reached out and poked an update directly.

Leo: So do we know how big, I mean, the botnet FBI said was 500 million routers at that time, and they told everybody to reboot your router, which obviously was inadequate. Do we know how big it is now?

Steve: I would imagine some probably fell off. The number I'm still seeing is more than half a million.

Leo: Half a billion. Oh, half a million, that's right, 500,000, right.

Steve: Yeah. It's still about 500,000. What we now know is there are additional stage three plugins. One in particular is...

Leo: Oh, man. This is a nation-state for you, right there, yeah.

Steve: Oh, yeah, yeah, yeah. This is like, you know, so...

Leo: Oh, man, it's like Stuxnet or something, yeah, wow.

Steve: So the stage three things are what live in RAM. And, for example, they found an HTTP man-in-the-middle interceptor so that, when a page is coming in that it is able to intercept, it drops the "S" from the HTTPS, turning everything into HTTP.

Leo: Oh, my god.

Steve: Yeah. I mean, so it's basically doing an HTTP security downgrade attack on anybody who's inboard of one of these affected and infected routers. So, I mean, it is clearly some serious intent when you find a module written for a particular make and model of VPN router which would not run on a different router. It means that someone, somewhere, wanted to target routers of that make and model, which happens to be a VPN router, in order to intercept its traffic and have access to it.

And as I was thinking about just sort of overall this podcast today, I was thinking about the world we're in today, which I've increasingly referred to as, 10 years ago even, this would feel like science fiction. And it was. It was the work of fiction not that far back that this kind of network - it was Daniel Suarez that was basically painting this kind of picture for us. And also Mark Russinovich has done the same thing, people who have extended what's possible.

And what I'm increasingly feeling, when we learn about things like little glitches in Zip, and these widespread problems in so many different makes and models of routers, one of the analogies that I've drawn on often is the notion of security being porous, that the barrier is not absolute. It's porous. And as you press on it more, it will spring leaks. You'll push molecules through this semi-permeable barrier.

And so what's happened is we've sort of gone along happily for a few decades, oh, look at the Internet, isn't it wonderful, and we can talk to everybody, and everybody can talk to us. And ooh, email, and ooh, messaging and web and wow, you know. But none of this stuff that has been built was, I mean, it was built to work, not to be secure. I mean, yeah, everyone said it was secure. Steve Ballmer was prancing around the stage, talking about how Windows XP was going to be the most secure operating system ever. I saw XP referred to online somewhere the other day as a "steaming stack of security vulnerabilities." It's like, ugh.

Leo: Although one could argue anything that's got 10 million lines of code is going to be a sieve; right?

Steve: Well, yes. And this is - well, but our philosophy, that suggests then, is broken.

Leo: Right.

Steve: That is, the development philosophy is, oh, just add more code to it. It's like, no. Don't add more code. So I really...

Leo: Also, I don't know if we anticipated these state-sponsored actors who were so adept and so well funded.

Steve: Yes. Well, and who could have anticipated malicious cryptocurrency mining?

Leo: Yeah, that's another one, right.

Steve: Or blackmailing people by encrypting their system's files. I mean, we've had like these weird things that have like, arose - arisen.

Leo: Arisen. Arise. Arose.

Steve: And, like, out of nowhere. And so because once upon a time cute little Word macro viruses didn't make anybody any money.

Leo: Right.

Steve: They were just sort of...

Leo: It's vandalism.

Steve: They were an itch.

Leo: It was because you could do it.

Steve: Yeah, exactly. Just like, you know, just like an itch. It's like, okay, go away, you annoyance.

Leo: And yet I think science fiction authors like Daniel Suarez probably could have anticipated this. And there were people who were saying, you know, cyberwarfare is going to be an issue. We're in it now, boy. And very vulnerable. What is this - you didn't mention the Picture of the Week. Is this related to VPNFilter?

Steve: No. It's actually a story later. That's what happens when, and we will get to it, when a vulnerability is found in a port that was mistakenly left open on Android devices.

Leo: Oh, this is the Android one, ooh.

Steve: Yes, yes, yes, yes. And it's not good. So anyway, I just, you know, VPNFilter is apparently - I wish it had a better name. You know, VPNFilter, that's just, you know, Heartbleed, that's something to get behind.

Leo: That's scary enough.

Steve: Yeah, exactly. Or Spectre or something. But VPNFilter? That sounds like something you want...

Leo: Yeah, yeah, it does.

Steve: ...rather than something you don't want. Going to filter my VPN, oh. Okay. So as if that weren't enough, Guardicore Labs has just published details of their discovery of something they've named Operation Prowli, P-R-O-W-L-I, which is a network of more than, yes, 40,000 compromised victim machines owned by more than 9,000 companies around the world, which is using some vulnerabilities in CMS, content management systems. And they didn't say Drupal specifically. Also, HP backup servers apparently have a vulnerability or a weak password. Also, believe it or not, DSL modems, I guess some of them are smart enough to be infected. So at some point you're just too dumb to have an infection. But unfortunately, DSL modems have gotten smart so they can be infected. And of course IoT devices.

So once again - and this doesn't feel state sponsored. They're mining cryptocurrency. They're redirecting traffic to promote fake websites and running tech support scams. So this just sort of feels like a larger independent group who have focused on a number of vulnerabilities. By watching the traffic, Guardicore basically knit an awareness that this was a single entity together because they spotted overlaps in the command-and-control servers of what would otherwise look like different groups operating independently. Because, for example, to attack these HP backup servers using weak passwords or a vulnerability is different from maybe using Drupal, the known Drupal vulnerabilities, and whatever it is that's wrong, probably an exposed management interface, on the DSL modems. Those are all different vectors.

But what they noticed was they're, like, all phoning home to a common set of IP addresses for command and control. So they said, oh, this is a bigger group than we thought. So they found it using various attack techniques, for example password brute-forcing, and weak and known vulnerable configurations. So just, again, this is one of the other things, as I was saying before, that just sort of, unfortunately, you can't not call this a trend. I mean, we're seeing this trend sort of emerging of this huge buildout of not fully secure Internet-connected stufferage, stuffage, stuff. And, boy, getting stuffed, unfortunately, is what's happening.

So again, I wanted to put this on our radar. We'll keep an eye on it, just in case it grows. At some point there really will be some turf wars over the available stuff. And the other thing that I think is going to happen, I refer to this a little bit later, or actually in the next story about Drupalgeddon2, is at the moment, as I've said before, we can take a little bit of solace in the fact that, when people get into a server, when the bad guys get into a server, their intention is to use its compute resource. Which is sort of like, okay, that's good because what you don't want is them turning around. You want them facing outwards toward the Internet. You don't want them turning around and looking inwards into your network, which they also do have now access to. It's like, okay.

So I have a feeling, before long, when maybe, who knows, it becomes too expensive even to mine bitcoin on stolen hardware, if that ever happens. But something could happen where this mass exploitation, this mass compromise of devices is no longer generating money enough that it keeps people from worrying or wondering, the bad guys, what's behind the servers that they're inhabiting because that'll be a whole 'nother problem.

So speaking of that, we have Drupalgeddon2, which of course it's "2" because several years ago was the first real problem with the Drupal Content Management System. And of course we've been speaking for the last few months, I think we're, what, three months in now from when the announcement was made on a Wednesday that the following Wednesday would be a big reveal, and everybody - the hope was of the Drupal devs that by putting out - like everybody should get ready. I remember you and I talked about it. And your TWiT crew, because you guys are Drupal users, your guys were already aware and ready to patch the moment this fix was released because the presumption was bad guys would be on it immediately. And within 24 hours, that following Thursday, indeed there were scans on the 'Net for this, and then a couple days later the beginnings of exploits. And now it's full blown.

So taking advantage of, like, a few months to give things time to settle, Troy Mursch at Bad Packets recently performed a passive scan of the Internet to get an appraisal of Drupal installation version numbers because versioning is one of the ways that we're able, well, passively to get some sense for what's going on, like are Drupal sites keeping themselves current or not.

So he wrote: "In my previous post," he says, "I detailed a large cryptojacking campaign that affected hundreds of Drupal websites." And this is just related to Drupalgeddon2, but not it itself. He said: "Multiple campaigns remain active today and are documented further in the latest SecurityTrails report. An important question was raised during my initial investigation: How many Drupal sites are vulnerable? To find the answer I began looking," he writes, "for sites using Drupal 7." This is, remember, there's 7 and 8, and then the really old ones are 6.

He said: "This is the most widely used version per Drupal's core statistics. Using the source code search engine PublicWWW," he says, "I was able to locate nearly 500,000 websites using Drupal 7." He says: "I promptly began scanning all the sites to establish which were vulnerable and which were not." He says: "I regarded sites that were using at least v7.58 as not vulnerable to Drupalgeddon2. This critical flaw is detailed in" - and this is the Drupal security advisory that is Drupalgeddon2, SA-CORE-2018-002, which has been assigned the CVE number 2018-7600. And you're showing now the pie chart on the screen of what he found.

He said: "Upon completion of the scan I was able to determine 115,000 sites were outdated and vulnerable." That is, older than the version of 7 which was the fixed one. 134,000 were not vulnerable, and 225,000 he was unable to determine one way or the other. He could not ascertain the version used.

Now, he notified the Drupal developers of this, and they were not happy with his assessment and responded to him, and publicly, that the number might be much lower than he's claiming, the 115,000. Well, of course they're wanting it to be true that the number is much lower.

Leo: Well, but his methodology was also pretty janky. I'm sorry.

Steve: Yes. He said...

Leo: If you look at our site, you can't see the changelog. I mean, come on.

Steve: Right. Well, but this is the changelog he did find.

Leo: We'd be in the red bar chart because they don't know if we've updated.

Steve: Correct, correct. But still, for 115,000 to be publishing their changelog which allowed him to determine what the version number was. But the developers commented that it was possible to remediate the problem without it being reflected in the changelog. So he of course, in order to go any further, would need to attempt an exploit of the vulnerability, which would be illegal, and so nobody but bad guys are able to get an accurate assessment for exactly what that count is. But if nothing else, it's a little worrisome that, of the versions which are visible, because the changelog would show the most recent version, 115,000 are older than the v7 which was fixed, which was 7.58, and are not being kept current.

So what we do know is that best practice says keep yourself updated to the most recent version. Even if you're not going to go to 8, for example, maybe compatibility problems, at least be maintaining currency. And it looks like there are - and it's not surprising. It should not surprise us because this is what we see, not only with Drupal, but pretty much with everything else, is that lots of exposed systems on the Internet are not being maintained.

Leo: Yeah. Drupal's a little bit of a different beast because there have been some very big changes between major version numbers on Drupal. And, for instance, we didn't update our site for a long time because Drupal 5 to 6 was a really...

Steve: Big change.

Leo: Yeah, you couldn't use your existing stuff.

Steve: It, like, broke everything.

Leo: Broke everything.

Steve: Broke everything, yes.

Leo: So there are a lot - there probably are a lot of Drupal 5 sites out there. Doesn't mean they haven't mitigated the flaw. They just haven't updated to Drupal 7 because it's such a big change. Drupal's unusual in that regard, I think, as content management engines go. For some reason they really make massive changes in between major version numbers.

Steve: Well, it's good to know.

Leo: I'm familiar with this only because we've been running Drupal since 2005. So I know how hard it is to do an upgrade.

Steve: Yeah. So, okay. Kevin Beaumont is a well-known security researcher who tweets from his handle @GossiTheDog.

Leo: Oh, wow.

Steve: Yup. He titled his blog post four days ago "Root Bridge" - and I was thinking to myself, or root canal. He said: "Root Bridge - how thousands of Internet-connected Android devices now have no security and are being exploited by criminals." It turns out the Android OS, which is used not only by smartphones, of course, but by cars and televisions and DVRs, and lots of people have taken Android OS and packaged it for their own purpose. It's a little general purpose operating system that can be useful for all kinds of things.

Leo: And it's free, and it's open source.

Steve: Yup, all the good things. Unfortunately, well, okay. Yes. Unfortunately, it can be misconfigured in an alarming way. There is a very handy feature which I actually have used myself in the past with an Android device known as the Android Debug Bridge, ADB for short, which allows developers to communicate with whatever the Android device is, typically over a USB connection.

Leo: Yeah. Everybody who's ever rooted an Android phone has used this.

Steve: Yes, exactly, exactly. It turns out, for example, the Android Developer Portal describes ADB, saying: "The ADB command facilitates a variety of device actions, such as installing and debugging apps, and it provides access to a Unix shell that you can use to run a variety of commands on a device." Okay, Leo. Now, imagine if that ADB interface, with all its power, were posted by mistake on port 5555?

Leo: That would be bad.

Steve: That would be bad.

Leo: That would be very bad.

Steve: That is what you call bad. Kevin explains: "ADB is completely unauthenticated, meaning anybody can connect to a device running ADB to execute commands."

Leo: That's true. There's no login. No, yeah.

Steve: Right. The presumption is, if you've got it, you have to have it in your possession in order to plug a USB connector into it, in order to plug it into your Mac or your Windows or your Linux box, in order to do whatever you want to do with it. Unfortunately, vendors...

Leo: Are stupid.

Steve: Get a load of this, have been shipping products...

Leo: Oh, lord.

Steve: ...with Android Debug Bridge enabled on WiFi, listening on port 5555. It's too bad it's not 666, but that's actually...

Leo: Why would they do that?

Steve: It's pure mistake. They had to have just, you know, the developer had it enabled in the config.

Leo: Yeah, they just never turned it off, yeah, yeah.

Steve: And they forgot, yes, they forgot, allowing anybody to connect over the Internet to a device. It's also clear some people are also rooting their devices, but then leaving - because you can, once you do this, once you get in, if you want the convenience of turning on ADB over WiFi, you can turn it on. But you certainly don't want to leave it on. Anyway, during the research that Kevin did, they found, as he wrote, everything from tankers in the U.S. to DVRs in Hong Kong to mobile telephones in South Korea. And, he said, as an example, a specific Android TV device which he has not named was also found to ship in this condition. He says in his blog it's highly problematic, as it allows anybody without a password to remotely access these devices as root and then to silently install software and execute malicious functions.

Leo: Oh, you have full access to the device. You can do anything you want.

Steve: Oh, yes.

Leo: You could read every folder. You could do anything you want, yeah.

Steve: Yes. So our Picture of the Week I repeated here in the show notes just to sort of

further drive the point home. This is what happened just after February 1st. Rapid7, the security project, their Project Heisenberg, and also GreyNoise Intelligence, whom we referred to a couple weeks ago, looked at what happened to port 5555 scanning. That's four fives. It just went vertically, like, straight up and has been high ever since. So what has happened is that an active campaign created an exploitable wormable piece of malware which is crypto mining.

Leo: Oh. Over that port.

Steve: Yes, yes. So whatever this device is gets infected. It begins mining. And then also, being a worm, it itself starts looking for other open port 5555s that it can infect. Which means we now have essentially a permanent presence of yet another thing on the Internet which will probably never go away completely. And if at any point a new Android device appears, if this mistake gets made again - and this is not just one device. That's what's weird is we've got tankers in the U.S., DVRs in Hong Kong, Android TVs.

Unfortunately, this appears to be, I don't know, it's probably not common, but it's not also impossibly rare. For example, in China alone, the use of Shodan, the Internet indexing vulnerability or open port indexing service that we refer to often, Shodan returned 82,274 devices in China that are accepting connections over port 5555. We don't know that they're all Android ADB, but that's the port that that service listens for connections over, unfortunately.

Now, I do have a takeaway for our listeners because the link in the show notes here, www.grc.com/x/portprobe=5555 will allow anyone to instantly check their device to see whether it is accepting connections over port 5555. And you've got that on the screen.

Leo: And I'm safe, thank god.

Steve: The good news is it's green, yes, you're stealth. You are safe, as you should be. Somebody with an Android device, for example, some smart device, you're welcome, obviously, to use GRC.com's ShieldsUP! port probe with a specific customized port in the URL that allows you to do pinpoint port probes immediately.

Leo: That's nice.

Steve: And, boy, if it comes up, what, blue, that would be - well, blue's not that bad. That's non-stealth. But if it's red, that means that, in response to GRC sending your device a SYN packet to port 5555, we received an ACK, meaning that the TCP stack in your device is saying, yeah, I'd love to talk to somebody on port 5555, which your device should not want to do.

Leo: You'd be protected if you're behind a router; right?

Steve: Yes, yes, yes.

Leo: I mean, most cases. And also this is probably not an easy thing to put on a device. You'd probably have to buy a device that already was misconfigured; right? I mean, malware...

Steve: Oh, yes. I'm sure, given the...

Leo: ...couldn't probably do this.

Steve: Yeah, given the number of devices which we're seeing exposed, it looks like it was just misconfigured out of the factory.

Leo: Yeah, yeah.

Steve: But you don't want one of those.

Leo: No. Thank you, I'll pass.

Steve: So what the heck has been going on at Cisco? Throughout this year Cisco has been performing some internal code auditing for which they should be encouraged and congratulated. And I've been watching these reports, and I haven't said anything because it's like, okay, I mean, I've been a little - I've wondered why Cisco keeps finding backdoors in their own products. That's a little worrisome. But it's not the end of the world. I mean, while it's disturbing that for some reason they're finding backdoors buried in their own products, at least they have the maturity and the foresight to be looking at their own code and not just assuming it's all fine.

And this actually was triggered, I think, by an earlier discovery by an outside party of a worrisome backdoor. And we did talk about that even further back. But now an external security researcher, Aaron Blair of RIoT, I got a kick out of that, RIoT as in R-I-O-T Solutions. He was researching a vulnerability in some Cisco software, their Wide Area Application Services, WAAS. And he leveraged a vulnerability that gave him access to the underlying file system on the platform he was using, which not even a normal device admin would get. With Cisco you could log in with extra, like, root administrative privileges, but that gives you more commands. You don't get underneath the OS in order to see the actual file system.

Well, this vulnerability allowed Aaron to do just that. And what he discovered was another previously unknown hardcoded backdoor, which he responsibly reported to Cisco, and they are fixing or have fixed. There's now an update for this Wide Area Application Services software. So that's good. And it even wasn't a really bad problem. In all of these networking devices there's a service called SNMP, Simple Network Management Protocol, which is a - I think it runs over port 161, if I recall. I use it a lot. Anytime you are monitoring, like, traffic on a remote device, you're sending SNMP, typically UDP packets, querying for specific counters to be told to you.

And they have a bizarre protocol. It's like there's a unified - they're called MIBs, M-I-B, which are sort of a dictionary of dotted numerical tree. So it's like 1.3.7.4.16.2., it's like

that. And at each dot is a multiway branch through this tree. And after about 20 of these dots, you finally get down to a leaf node where is a counter, which is like bytes in or bytes out or packets in or firewall firings of this packet or whatever. So it is cool because it allows - it's a standardized mechanism for allowing remote over-the-network monitoring of devices.

Now, write access is significantly more dangerous because it's possible also to configure devices over SNMP, if you have write access. For example, that tree, benign as it is, allows you to do things like add and remove filters and rules and NAT mappings and so forth. So it can be powerful. Still, nobody who's concerned about security wants somebody remotely, without authorization, to access the entire SNMP statistics tree of any of these devices where this WAAS software is. And who knows what other devices may also have this secret. So the big problem is here is like the fourth in just a few months of discovered hardcoded backdoors in the devices of a major, like the major - I mean, there's a lot of competition now, but used to be Cisco was it - Internet big iron hardware manufacturer with routers and switches and so forth.

And so, if you step back, you just sort of have to say, as I did at the top of this, what the heck has been going on at Cisco? I don't want to do the conspiracy theory thing; but of course in the post-Snowden era, where we have seen clear evidence of prior involvement by the NSA, their fingers seem to be in these things, and the idea that employees could be implanted in corporations or turned once they're there or believe that they are supporting U.S. domestic security by just putting a cute little backdoor into something. The problem is, as we know, a backdoor of this nature can be used by anybody who knows about it.

And so here's Aaron discovering this because he gets access to the file system through a vulnerability, which he then reported responsibly to Cisco. But then he also had to say, oh, and by the way, I found an undocumented backdoor in your SNMP service which you might want to look at, too. So of course it begs several questions. Why are there backdoors that it sounds like even Cisco themselves do not officially know about? How long have they been there? Why are they there? And which other ones are there that aren't known? And this clearly got onto Cisco management radar somewhere because it had to have come, apparently it came as a surprise to them, too. And so they started performing an internal code audit of their own code to figure out, okay, we can't deny the truth any longer, that somebody has been putting code in our own products. So, yikes.

Leo: Yeah, I'd love to know who. Wow.

Steve: Yeah, yeah. I mean, and these are, unfortunately, the things we will never have answers to. But still, it's worth proceeding with caution.

I guess it was inevitable, I'm sure it was inevitable that cryptocurrency miners, which want to mine on consumer devices, would attempt to become stealthy. And they have a problem because what they want to do in order to mine is use resource, CPU and hopefully GPU, when they can get it. Yet if people are using the computer, like for their own work, especially a gamer who intends to saturate both the GPU and the CPU, they're going to notice if their frame rate suddenly falls to 2, or if the game just stops playing the way they're used to it. Or if somebody who is somewhat savvy realizes that their computer has suddenly gotten sluggish, then, I mean, I know that I, and I know you and other people, Leo, will fire up a task manager, a task viewer, to see what process in our system is hogging all of the memory and/or, you know, sometimes it's memory. In this case, with a crypto miner, it's CPU resource.

So Bleeping Computer's Lawrence Abrams described a new miner which they had become aware of which is doing exactly this. Now, it's not particularly clever. I'll describe the clever solution in a minute. But what they're doing is - so this thing gets into someone's machine. It establishes an entry in Task Scheduler so that, at midnight, the Task Scheduler will first trigger. So it doesn't do anything when it first gets into the machine. And using Task Scheduler is a time-honored means for malware to execute itself because Task Schedule is able to run things for which it has been set up to do so.

At midnight, the Task Scheduler triggers and then repeats every minute. So that launches the miner, which checks to make sure that there isn't already an instance of it mining; and, if so, then the newly relaunched instance immediately terminates. While mining, it proactively enumerates the system's process list in order to - and this is what I'm thinking is, well, okay, this is kind of brute force and not really very clever - to look for instances of Process Explorer, Task Manager, Process Monitor, Process Hacker, and the AnVir Task Manager running in the system. It infers, if it sees those running, that a savvy user may be wondering what the heck is going on. And so if it sees those, it terminates itself immediately, disappears.

It also looks for, and this is a weird list, but for Counterstrike: Global Offensive, PlayerUnknown's Battlegrounds, Rainbow Six, or Dota 2, which are games which want to have full use of the system and where, if you've got a crypto miner trying to also share the system, that game is not going to run as well as it should. If any of those things are found, it terminates itself immediately in order to keep from being more permanently removed from the system. And then it knows that Task Manager will continue to do its job, which is to trigger a reawakening every minute. Presumably the first thing it does before it starts to mine is do this process enumeration to see whether any of these things are present; and, if so, it terminates again.

So this just seems, I mean, it's like, okay. We've now entered a world where cryptocurrency miners have realized they're giving themselves away if they just sit there and squat on a person's machine. Their strategy will be much more successful if they arrange not to cause themselves to be deleted. So as a developer...

Leo: It's called "nice" mode. It literally is; right?

Steve: Exactly.

Leo: Nice.

Steve: Right. So, well, yes, nice in Unix, exactly.

Leo: Yeah, yeah.

Steve: What I'm a little bit surprised about is that whoever it was who did this didn't do it cleverly. So, for example, what I would do if this were my focus, and it's not, but first of all I would reduce the process's own priority to the lowest possible so that it essentially uses the idle time of the system and gives up all processing preferentially. Then I would, if I were a malicious crypto miner, monitor my own hashing rate, which I

can certainly do, to notice if the hashing rate drops because that would tell me that this system is not idle, it's in use, and that I should consider fully backing off and pausing all mining and then just checking from time to time. I mean, it's easy to determine whether the system is idle or not.

So watch the system, wait for it to go idle, then go in with low priority background task, which you could have 100% of the system. You just can't have it if anything else wants 100% of the system. In which case you get to mine, you're not in the way, you are probably going to get to stay around a lot longer because you're not going to cause somebody to go run Malwarebytes or whatever on your system in order to spot and remove you.

And I think what we're going to see - because it's very clear that crypto mining has supplanted the previous file encrypting malware as the preferred means of raising money because lots of people had backups, or they objected to paying, or they didn't have anything valuable on their system. Well, the actual presence of their system is now becoming valuable. So it'll be interesting to see where this evolves. I think we're going to be seeing some evolution.

I did want to mention that, as I talked at the top of the show, that Palo Alto Networks has a group they call Unit 42 which is kind of fun because I'm sure 42 came from Douglas Adams. That's of course the ultimate answer to all things is 42. Russia's Fancy Bear, APT28, this Sofacy, I mean, they're known by many names. They've been observed to have changed their tactics from a highly targeted, go after a few people in an organization with phishing attacks specifically designed to stay under the radar. They're now doing more of a scattershot. Palo Alto Networks Unit 42 group has noted that these attacks from this Russian state-sponsored group appears to be using email addresses easily found with search engines to spray organizations that they want to get into.

And the first thing that occurred to me is to wonder whether maybe the group is under new management. I mean, literally. I'm not kidding. Like somebody just decided, okay, we're going to take a different approach, and so said to the techies, let's try something different. There's a lot more detail in the report that I don't want to get into. But it did occur to me that it's interesting that we're seeing a shift from what has been a well-proven strategy to something that they described it as a focus-fire-style attack in the past to something much less focused and a little more impatient to get inside of organizations.

I've talked also about a genealogy site, MyHeritage, which had lost control of their users' login account data. And of course when you hear that a site to which you have uploaded your DNA has lost control of their data, that would be a privacy concern for some people. The good news is there are apparently three entirely separate databases. There is the user login account database. There is the financial credit card charging-people-for-the-service database. And also, separate from either of those two, is the genealogy we've-got-your-DNA-that-you-uploaded-to-us database. All are separate. And they did lose control, they discovered, when someone found the archive of 92,283,889 users on an online archive. What was there was email addresses and password hashes.

To MyHeritage's credit, in addition to having all of the databases separate - which is good security policy, so that if this happens it's not people's DNA; you don't have to tell people, gee, we're really sorry, but we leaked your personal and private DNA data that we promised we would keep secret. Also they disclosed it the day of the breach, that is, well, the same day they learned of the breach. The breach occurred on October 26th of 2017. They know because that was the latest date found in the archive. And so that was when the snapshot was made of the database. And they have employed a forensic

security firm to dig in further and see what more can be learned about what happened.

The other good news is, and more reasons to give them some credit, is not only did they immediately disclose the breach, but the passwords were hashed with a per-user salt. So if they used per-user salt, although they didn't specifically say in their sort of generic disclosure, it's very likely they also used some sort of PBKDF, a Password-Based Key Derivation Function, meaning it wasn't just a salted SHA-256 where they used a random salt, but they may have iterated that some useful number of times to make it very difficult to reverse the hash back to a password. So the per-user salt, of course, means that they're preventing any sort of massive gang reversing of all 92-plus million users, which could have been done if they'd used a single shared salt.

The good news is it really looks like these guys have their security nailed. They also indicated that they would soon be deploying second-factor authentication to further strengthen their users' login safety. But when this happens, when password hashes escape, standard advice is you've got to change your passwords as soon as possible. So even though you're probably safe, anyone hearing this who didn't receive notice already directly from MyHeritage, hopefully you already know this, you do want to change your password.

It's also worth noting that another possible reason for their speedy responsible disclosure could be chalked up to the EU's GDPR regulations, which as we have been discussing are now in place. And the GDPR requires companies doing business in the EU to disclose within 72 hours, that is, three days, of learning of a breach which affects the privacy and data of EU citizens. So I think we're going to see a change of behavior. We're not going to see anymore of this Equifax nonsense where, oh, yeah, we knew six months ago, but we didn't get around to telling anybody about it. So anyway, these guys look like they've got security in place. They did a good job. And hats off to them.

Leo: Wow, good news.

Steve: Yes. Okay. Adobe Flash cannot die soon enough. Once again, an actively exploited in the wild zero day had come to life. They announced an emergency patch last Thursday. I don't know. Like in other words, when malware - of Flash you would have to say "when malware has a large, reputable publisher." I mean, that's what this is. This is just ridiculous.

Okay. So Adobe's own summary for this update said: "Adobe has released security updates for Adobe Flash Player for Windows, macOS, Linux, and Chrome." Even Chrome's built-in Flash player was vulnerable. "These updates address critical vulnerabilities in Adobe Flash Player 29.0.0.171 and earlier. Successful exploitation could lead to arbitrary code execution in the context of the current user. Adobe is aware," they write, "of a report that an exploit for" - and then there's a vulnerability for their Flash Player - "2018-5002 exists in the wild and is being used in limited, targeted attacks against Windows users. These attacks leverage Office documents with embedded malicious Flash Player content distributed via email."

In some of the coverage of this, Leo, I saw some observers saying, well, browsers have hardened themselves proactively against Flash, so now the miscreants have figured out another way to leverage the continuing existence of Flash, even though browsers won't use it any longer, but it's still present in people's systems in order to exploit them. And look at this exploitation path. I have the picture here in the show notes. It's like, yikes.

The user opens a weaponized Office document which contains a Flash ActiveX embedded object. That reaches out to a remote server to obtain a Flash exploit which is retrieved using public key crypto and then executed, which decrypts itself and then, in Step 7, uses an exploit in Flash, Loader.loadBytes, to load and trigger the exploit, which then reaches out again to the same server to download shell code, which it's then able to execute on the user's system.

So remind me again why it is exactly we're waiting until 2020, the end of 2020, through 2020, to finally be rid of this Internet menace called Flash. I mean, we have routers infected, half a million routers infected with self-destructing firmware, ready to be triggered on word from Moscow. Yet how difficult could it possibly be for Adobe to push out just one last update to self-destruct all remaining instances of Flash in the wild and make everyone safer? It's just like, my goodness. It's only now being used by websites that still want to use it to display video. Yet we've seen that this slow end-of-lifing doesn't work on the Internet. I mean, we've covered multiple instances where Chrome, it's only by refusing to allow things to run any longer, and creating well-published deadlines, that people have been forced to move away from technology that, well, you know, it's working, so let's not change anything. Except its presence there is hurting everyone.

So anyway, I just had to rant again on another occasion of a zero-day being exploited, of Flash, by virtue of the fact that it still exists in people's machines. And what Adobe is going to have to do to be responsible, I mean, I guess we're going to have to wait until the end of 2020. But they need to have it self-destruct. If they're saying we're going to no longer support it, then please, Adobe, kill it. Shoot it in the head. Just end of Flash. Goodbye. That's what we need.

And, finally, checking in on Marcus Hutchins. I didn't dig into this in depth because there's just too much of this is opinion. And you can find - it's the Internet; right? So you can find any opinion that you want about anything on the Internet. But a pro freedom of the press blogger who has her Ph.D. in something, I don't remember, I mean, I read her CV trying to figure out who this is. An independent journalist is of the opinion that the government has always been on shaky ground with Marcus. I mean, you and I have talked about the plight that he's in, that it looks like in the past he did some things when he was a kid that he's not proud of and that he would take back if he could, that he probably regrets. And of course, as we know, he was trying to leave after the last summer's Defcon and had to go to the U.K. and got nabbed by law enforcement in Las Vegas, and has been under some form of arrest ever since while a jury trial has been moving along slowly.

And I was surprised to see that. I mean, I guess it must be that the prosecution felt that they would have a better case with a jury than with a judge. But some of the opinion that I've seen suggests that the government's case is beginning to fall apart, and that there may be a dismissal, a full dismissal of the case against Marcus pending.

Leo: Interesting. Hmm. Wow.

Steve: Yes. Which, you know, we could hope for. The title of this update was "To Pre-empt an Ass-Handing, the Government Lards on Problematic New Charges against MalwareTech," and that's of course Marcus's handle. This journalist wrote: "But the government, which refuses to cut its losses on its own prosecutorial misjudgments, just doubled down with a 10-count superseding indictment." That is, in many cases supersedes the previous counts of indictment. "Effectively," she writes, "the superseding

creates new counts, first of all, by charging Hutchins for stuff that, one, is outside a five-year statute of limitations; and, two, he did when he was a minor, that is, stuff that shouldn't be legally charged at all; and then adding a wire fraud conspiracy and false statements charge to try to bypass all of the defects in the original indictment."

And she writes: "The false statements charge is the best of all because, for it to be true, a Nevada prosecutor would have to be named as Hutchins' co-conspirator because his representations in court last summer directly contradict the claims in this new indictment." So again, I haven't read into all this. I don't know. I'm not an attorney, so I can't render an opinion. But it's nice that it's not just a slam dunk, you know, you're guilty and we're locking you up forever, thank goodness.

And we'll keep an eye on this. It would be wonderful if in fact this, I mean, I did read much more about this than I have said. But again, I'm unable to render an opinion one way or the other. We can just hope that in fact this is going to fall apart. It does look like these new charges, they found some other piece of malware which, if the timeline is correct, he would indeed have had to write it when he was not yet 18, and the statute of limitations would have expired since then. So we'll see.

And lastly, before our final break, I'll share a fun story from Mark in Merced, who sent this on the 2nd of June. The subject was "Fun SpinRite Story." He said: "I wanted to share a quick story with you. My wife was using our household laptop when it started to lock up and display a 'Wait/End Task' message when opening the Start menu and doing other things. So," he writes, "I ran SpinRite on the machine. It found a few bad sectors on the hard drive, and I said 'Aha' to myself.

"Needless to say, the laptop now works great again. So I hand it back to my wife, and she gets back to work again on a Photoshop project. A bit later, she wants to show me something she was working on before the incident. So that I can get a better look, she goes over to the coffee table with the laptop sitting on top of it and drags the table my way on the carpet. I watch the laptop jiggling, bouncing, and jarring on the table top and think to myself, 'No wonder the drive needed SpinRite.'"

And of course this has been a longstanding observation. Hard drives are serious high technology. I mean, it's astonishing to me today that we have the data storage density that we do. And so it means that these devices are fragile. And I can't think of a more hostile environment, okay, except arguably an external USB plugged-in drive, that's an even greater danger, because everyone's heard of a head crash. It literally means the heads crash into the disk because, other than the Iomega devices, those ZIP drives that were so famous, and the Bernoulli boxes back in the day, heads are not in contact with the magnetic surface. The spinning pulls air underneath the head to create essentially a cushion, and the head is being pressed down hard against this air cushion, but it is flying very closely, yet not in contact with the magnetic surface. And this is the hard drive technology we've had for many decades. Consequently, this is fragile.

And, boy, you want a hard drive mounted in a server, bolted to a rack, bolted to the ground, in an area with no earthquakes, in order to keep it immobile. And we've seen that even screaming at a hard drive can cause it to have problems; or when the fire suppression system whistles loudly, that can cause problems. So, wow, I can certainly understand what Mark meant when he said he watched the laptop bouncing on the tabletop and just sort of shook his head and thought, well, maybe I'm going to need to explain to my wife how delicate this is and run SpinRite again before there are problems that can't be fixed. Wow.

Leo: Well, there you have it. It's good he had SpinRite. That's all I can say.

Steve: Yup. So Zippity Do or Don't. Boy. Okay. So as I said at the top of the show, one thing we keep hitting is how difficult interpreters are to get right. I mean, and for a good reason. Whether you are decompressing a JPEG, or you are re-rendering a Java byte stream back into a Java object, or you are decompressing a blob into its original form, the way this is done is instructions are read, little tokenized - it could be just a combination of bits which has a larger meaning. And so if it's 001, that means go over in this direction and then look for further instructions. If it's 010, then go in this direction and then see what comes next. I mean, and this is the way we decompress or we reassemble or we expand something from an encoded form to the decoded form.

The problem is it is so sort of implicit to assume that the encoder produced what we are decoding. That is, we assume a benign source of the encoding which we are then going to decode. It's difficult for me to deeply enough articulate how hard it is to break that assumption. You invent this amazingly cool compression, and you're proud of it. And then you write the decompressor that reverses the compression, and you go, look, I got back what I put in. Isn't that cool? Or in the case of JPEG, I got back something that looks the same as I put in, even if it's not the same. Which is kind of even cooler, when you think about it. It's like, my eyes can't tell the difference; but look, you know, it looks the same. It's a photo.

So it turns out that yet another rock got turned over recently with Zip compression. And we were just referring to it. We were talking about Huffman and Lempel-Ziv, the original patent holders at IBM in 1977 - it's a patent I studied extensively years and years ago - came up with this cool solution for - and we did a podcast on it quite a while ago, how Lempel-Ziv compression works. But fundamentally it's an interpretation. And just this week several different types of problems came to light.

Back two months ago, in April of this year, some researchers at a British software firm, Snyk, S-N-Y-K, named a discovery of a problem "Zip Slip." And they began informing users of a large array of compression libraries that they had discovered a widespread flaw. This affected RAR, which is hugely widespread; 7z, which is the 7-Zip compression, also very widespread; tar; jar...

Leo: Tar, really?

Steve: Yes, tar.

Leo: Uh-oh.

Steve: I know. War, cpio, and apk. And, I mean, yes, tar is, like, universally used in Unix and Linux systems.

Leo: Well, apk is the app file format for Android.

Steve: Yes.

Leo: Yikes.

Steve: Yes. So as a consequence, literally thousands of projects written in programming languages including JavaScript, Ruby, Java, .NET, and Go, published by everybody - Google, Oracle, IBM, Apache, Amazon. They listed Spring Pivotal, LinkedIn, Twitter, Alibaba, Eclipse, OWASP, ElasticSearch, JetBrains, and more. All contained vulnerable code and libraries.

Leo: Ai yai yai.

Steve: Yeah. Now, it can be exploited using a specially crafted archive file that holds - get this, Leo - an old friend of ours, directory traversal filenames. When extracted, any vulnerable code or library would allow attackers to unarchive malicious files outside the folder where it should reside. Now, we should stop for a minute and explain directory traversal. Also back from the dawn of time, in a filename or file path name, "dot" has referred to the current directory, and "dot dot" has referred to the parent directory. So, for example, I'm often, if I have, like, /asm/sqrl, and I want to go to /asm/ne, which is GRC's net engine, I could do a "cd /asm/ne," but I often just do ../ne, meaning move up one level in the hierarchy and then down to a different branch from the parent.

Leo: Do that all the time.

Steve: Yup, exactly. It's very handy, especially if you're way down in a hierarchy, like nine levels down, slash this, slash that, slash something else. You don't want to restate that whole deep hierarchy in order to just change to a different leaf at the same level. So you do ../ and then rename the leaf, and you're immediately there.

Well, it turns out that that little shortcut has been for decades a source of vulnerabilities because you can repeat that. You can go ../../../../../, and each of those moves you back up the hierarchy toward the root. And once you finally hit the root, redundant dot dot slashes don't hurt you. So you can just do 10 of them, and pretty much...

Leo: Oh, it'll always get you to the root.

Steve: Yes, it will always bring you back to home base. Now that you're there, first of all, that's a scary place to be because the root is the root, and you can then navigate back down a different path to somewhere else. It turns out that all of those unarchiving libraries were not protecting against directory traversal, meaning that, if somehow they were asked to unarchive a file within the archive that contained a long directory traversal exploit, they would do so.

And the archive itself contains exactly that. It contains essentially relative filenames that are relative to where you're unzipping. So like you say, okay, I want to unzip this to this file, to this folder. Well, you assume nothing can go outside of that folder. It's going to be there and deeper, not waltz itself back up to the root of your directory and then dig back down into /windows/system32 and replace a DLL, for example, or rewrite a configuration file somewhere that it shouldn't. But in fact all of these libraries can, at the time that this

was discovered, be abused in this way. Now, many of them...

Leo: You can't write to the root directory without admin permissions. So this is not an escalation attack. You'd have to have access.

Steve: Correct, correct. Although what they found was many places where this could be leveraged as part of that, or that you're able to rewrite some configuration file which does not itself look like it's a problem, but can then be used in order to stage a larger attack.

Leo: Right, right. And often people use these commands as you do escalation.

Steve: Exactly, exactly. So for the last two months Snyk has been quietly and privately disclosing what they call the "Zip Slip" vulnerability. Of course it's named that because you're able to slip backwards up the file system and then back down to all the vulnerable libraries and project maintainers that they were able to find to give them a chance to update this. And of course the trouble is this is such a longstanding problem, and so far and widespread that it's going to continue to exist for quite some time. So as I said at the top of the show, it's sort of scary that we're finding as many vulnerabilities as we are in systems that are as widespread as we are using them.

So in their posting they propose the question: "Are you vulnerable?" And they write: "You are vulnerable if you are using a library which contains the Zip Slip vulnerability, or your project contains vulnerable code, which extracts files from an archive without the necessary directory traversal validation. Snyk is maintaining a GitHub repository listing all projects that have been found vulnerable to Zip Slip and have been responsibly disclosed to, including fix dates and versions. The repository is open to contributions from the wider community to ensure it holds the most up-to-date status."

So in the show notes I have a link to - there is a PDF with a full technical whitepaper, and then it's Snyk.io, S-N-Y-K dot io.

Leo: I think it's Snyk, not Synk.

Steve: Oh, it's Snyk?

Leo: Snyk.

Steve: Okay, Snyk. The Snyk Zip Slip, yeah.

Leo: Here's all the libraries. They have a GitHub list of all the libraries, wow.

Steve: Yup. And it is extensive.

Leo: Package manager, Java, .NET.

Steve: Yeah. They noted a bias. When they sort of stood back, they noticed that some packages tended to be, or some languages tend to have less instances of vulnerability. For example, they said that Go had fewer because it tended to provide the services natively. Whereas, for example, Java had more problems because Java doesn't have a central library offering high-level processing of archives, for example, zip files. And they wrote: "The lack of such a library led to vulnerable code snippets being handcrafted and then shared among developer communities such as on Stack Overflow." So there did tend to be some sort of a bias in general. But overall, you want to make sure that you're safe from this.

And so I will conclude with one more, believe it or not. There was a problem found in another series of rar and zip instances which affected AV. Avast, Bitdefender, and F-Secure were all found to have problems with their unzipping. We've talked about the attack surface vulnerabilities created by AV. And in order for AV to do its work, it needs to look inside the compressed archives that you may be downloading, if they're a zip or a rar or whatever, a 7z, in order to see what's inside them because they're often used to obscure malware of various kinds.

But all of these decompressors are themselves interpreters, and they have had a series of flaws that could be deliberately abused. In the case of rar, which was the most recently fixed, which affected F-Secure, there's something known as "solid mode." In solid mode - and I've used it myself because you get a denser compression if you encrypt with solid mode. Solid mode prevents you from afterwards editing the rar to, like, add or remove individual files. So it's the sort of thing you want to finally only do when the rar is going to be, as the name suggests, solidified and then no longer treated as sort of a quasi file system that you could add and update and remove things to and from.

Well, the reason is that, if you don't generate a solid rar, every file resets the state of the compressor to its initial condition so that each file starts being recompressed with no knowledge of the past. It is the brilliance of the Lempel-Ziv compression, which all of these things use, where the history of what it has just recently seen informs it about what it may be seeing in the future. And by simply pointing to what it has just seen, you're able to eliminate the redundancy and then achieve compression. So if you compress in solid mode, then the state of the compressor is not reset at the beginning of each file inside of an archive, which gives you greater compression, assuming that a run of files is going to be largely the same kind of information, like a bunch of source code or a bunch of exes or whatever.

So it turns out that they did not check, that is, the rar unpacker did not check to see whether it had ever encountered a non-solid flag in a file. Which is to say, the first file in a rar should always say "initialize the state of the decompressor." Subsequent files could say "don't bother initializing the state of the compressor because I'm a follow-on file, and I want to take advantage of the state which already exists." So consequently the mistake was made that, if the first file in the archive claimed to be a follow-on file, then the decompressor state was not initialized. And an uninitialized buffer, especially when it's going to be interpreted by an interpreter, is a huge opportunity for exploitation.

And it turns out it was possible in order to leverage that into a remote code execution. Bottom line was, if someone knew that you were using F-Secure's AV, if they simply, in any way, caused your F-Secure AV, which I should mention has been fixed since, so you want to make sure you're up to date because it was just recently fixed, if your system

touched one of those files, it could take over your computer.

So again, another instance of an interpreter where the designers knew what they were doing, but they failed to look at the concept, failed to take into account the ways in which the meta language that they produced as a consequence of compression could be abused by an attacker. And this was another clever example of that happening, which you really don't want to happen, you don't want to have in your AV because anything coming in through any external mechanism into your computer could trigger the AV which is trying to scrutinize everything. And if you've got problems there, your machine can be victim. So once again, another instance of an interpreter gone bad, or a weak, insecure interpreter biting us.

Leo: I guess you could call Flash an interpreter.

Steve: Oh, it's an interpreter from hell.

Leo: I mean, a lot of the exploits we see are interpreters. And the reason that's worrisome is because it means a document file, which normally would be benign no matter what, can be used to attack you if you have an unpatched interpreter, as it were.

Steve: Yup.

Leo: Steve Gibson has done it again, has he not? A fun two hours, thank you, Steve. We do Security Now! every Tuesday about 1:30 Pacific, 4:30 Eastern, 20:30 UTC. If you want to stop by and watch it, you can, the live streams. And they are moving around a little bit. We've got some new ones. I think we're saying goodbye to some old ones. But TWiT.tv/live should always have a good choice for you, both audio and video live streams, if you want to watch live. You can also come in the studio. We have some nice people in the studio audience today visiting from Atlanta, Georgia, and San Ramon, California, just up the road apiece. Thank you, Matt and Joan and Paul. And if you want to do that, just email tickets@twit.tv, and we'll put a chair out for you.

If you want to download on-demand versions of the show, Steve has really a great place to start at his website, GRC.com. Not only can you download audio of the show, he also does transcripts. So that's a way you can read along with the show as you listen or after you listen or before you listen or you never listen. You can just read it. We have audio and video at our site, which is TWiT.tv/sn. And of course every podcast app carries Security Now!, so you could subscribe in those. When you're at GRC, don't forget to check out Steve's bread and butter, his great, must-have hard drive recovery and maintenance utility, SpinRite. You should also...

Steve: And if you're jiggling your laptop around, then keep an eye on it, for sure.

Leo: You know, that's one of the reason I like solid-state drives. There's no moving parts.

Steve: Yes, yes.

Leo: They really are more robust.

Steve: And the good news is SpinRite fixes them, too.

Leo: Yeah. From other things. GRC.com. Steve, we'll catch up with you next Tuesday.

Steve: Thank you, my friend. Always a pleasure, thanks. Bye.