SECURITY NOW!

Transcript of Episode #778

BootHole

Description: This week we touch on the recent update of Firefox to v79. We check back on the Twitter hack with the news of the identity of the accused perpetrators. We have more information about the Garmin ransomware hack. We look at the behavior of another disgruntled vulnerability researcher and consider another aspect of the ethics of vulnerability disclosure. We examine Zoom's bug of the week and the consequences of Microsoft's removal of all SHA-1 signed downloads, and note that QNAP NAS devices are still suffering from real trouble and neglect by their owners. I'm going to check in with the SpinRite work. Then we take a look at the week's biggest security event - the discovery of a boot security bypass for Linux.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-778.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-778-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Firefox 79 is also here. Steve has some new features. We'll talk about the Twitter hack. They got 'em, three kids, one 17 years old. We'll also talk about a real problem with security with Tor that they don't really even seem to care much about, and a flaw with GRUB2. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 778, recorded Tuesday, August 4th, 2020: BootHole.

It's time for Security Now!, the show where we cover the latest in the security world with Mr. Steve Gibson. Hi, Steve.

Steve Gibson: Hey, Leo. Great to be with you again for our "closing in on the end of year 15" Episode 778. And I think it'll be 780. I think about two weeks from now is when we lap ourselves, closing out Year 15 and beginning on 16.

Leo: Now, forgive me, but I told my team Steve doesn't want a cake or balloons or confetti or anything like that.

Steve: Correct, yes.

Leo: Steve's like me. We don't celebrate these anniversaries.

Steve: My 65th birthday occurred toward the end of March in the middle of COVID land.

Leo: We didn't even know.

Steve: And I thought, this is the weirdest birthday ever.

Leo: Yeah. Yeah, mine's in a couple - a year from November. Wow.

Steve: You are correct. No cakes, thank you very much.

Leo: No cakes, no more, unh-unh. Let's just pretend nothing's happening.

Steve: So we're going to talk about BootHole, which is a little reminiscent of Spectre and Meltdown inasmuch as the cure is in some cases worse than the problem, as we'll see. But we're going to start by talking about the recent update to Firefox 79, one sort of okay new feature, not much else changed. We're going to check back on the Twitter hack with news of, well, with news of the identity of the accused perps. And that's "alleged" identity because we don't know for sure. The so-called "mastermind" is a juvenile, so we're respecting that.

We also have more information about the Garmin ransomware hack. Some additional information has come to light, primarily thanks to BleepingComputer's digging and their access to some insiders who've been feeding them some interesting tidbits. We're also going to take a look at the behavior of another disgruntled vulnerability researcher and consider some aspects of the ethics of vulnerability disclosure that we haven't really talked about before.

We're going to examine what has now become Zoom's bug of the week, and the consequences of Microsoft's removal of all SHA-1 signed downloads from their site. Unless you're a bit of a packrat, that could pose some problems. Also QNAP NAS (Network Attached Storage) devices, are still suffering from escalating real trouble and neglect by their owners. We'll touch on that. I'll check back in with a little bit of update on my ongoing work with SpinRite and sort of deal with some of our listeners' questions quickly.

And then we're going to take a look at what is arguably, or I guess inarguably? I don't know. No one's arguing about it, the week's biggest security event, which is the discovery of this boot security bypass for Linux, which was named BootHole by its two discoverers. Oh, and just an interesting Picture of the Week that I just - I don't want to belabor the point; but, boy, it's a rather compelling graph that I wanted just to make sure our listeners were aware of. So I think another great podcast for our listeners.

Leo: As always, Mr. Steve Gibson. And I like this Picture of the Week. This sounds really good.

Steve: So our longtime listeners certainly know that I sort of glommed onto the potential importance of Vitamin D years ago. I surprised you one Tuesday, or maybe we were back recording on Wednesdays then. I don't remember.

Leo: Probably, yeah, yeah.

Steve: By, like, saying, well, we're not going to talk about security. We're going to talk about something different. And anyway, of course that was the now famous Vitamin D podcast [SN-209] where I went through all the biology and what appears to be the nature of the importance of it. I also talked about it early this year, at the beginning of COVID, because it's known to have an important intersection with immunity. And this is a year when we need all the immunity we can get.

The problem, of course, is that it almost costs nothing. A strong proper dose for a year for a person is like $15 for 360 "little drops of sunshine," as I call them. Anyway, the point is that, because it costs nothing, it's difficult to get money for research.

Leo: No big drug companies getting rich on this, yeah.

Steve: Exactly. This is, yeah, you're not having the White House funding the production of Vitamin D. We already have, you know, you could argue you get all you need from the sun, but there actually is some correlation between the amount you have in your bloodstream and your proximity to the equator. So that's one factor.

Leo: That makes sense, yeah.

Steve: But 20 different groups of patients who, as a consequence of this past year of COVID-19 health crisis, happened to have their serum Vitamin D levels tested, were pulled together in sort of an ad hoc study. I have the link in the show notes to the source PDF of which this is just one of many charts. But this is the most dramatic of them. This shows their correlation - and as we know, correlation is not causation, but this is, if nothing else, eye-opening - that correlates from these 20 studies whose results were aggregated.

It shows first of all in this sort of bluish line that there was no age difference over the span of Vitamin D concentration. It's pretty much 60 to 65 years of age, regardless. So age wasn't a factor here. But the red line shows a breathtaking correlation, given all the caveats, studies, varying levels of control. Obviously this is not a random sample. These are people that were already in trouble such that they had their blood drawn and so forth. So with that understanding, showing a very clear connection between the measured Vitamin D concentration and their ultimate consequences, that is, in this case the COVID-19 death rate as a function of Vitamin D.

So enough said. Vitamin D is good. Do your own independent research. If you're interested, click the link to the PDF. All the information about how this data was gathered is there. And at the end is a whole bunch of additional backup material from the NIH and other health organizations to substantiate it. So just wanted, again, not to spend any more time on this, but this was very powerful information. For what it's worth, I just didn't want it to go unobserved.

Leo: And what do you do? How much Vitamin D do you take?

Steve: 5,000 IU a day.

Leo: That's what I'm doing, too, yeah.

Steve: Yup. I think that, you know, if you're uncomfortable with that, maybe 4,000. But the RDA is 400, and that just doesn't get you off the ground. That's not going to do it. I mean, it will prevent you from dying of scurvy and some sort of acute Vitamin D shortage. But it's very different, I mean, and that's one of the problems with the RDA is it was - we call it the "Recommended Dietary Allowance." It was meant to be the minimum you need of different things to keep you from having disease as a consequence of a shortage. There's a big margin between what's enough to keep you from dying and what you should have for brimming health.

Leo: There's always, we've always known Vitamin D was, you know, you get rickets if you don't have Vitamin D. That's why they put it in milk as a supplement.

Steve: Correct.

Leo: So we've always known that. But there seems - go listen to Steve's whole show on Vitamin D because it really is an eye-opener. And this isn't the only place I've seen this kind of indication that having a good Vitamin D level, not a deficiency, anyway, is important for COVID survival.

Steve: Right. And in fact I'm glad you said that because I am, you know, I'm not wanting to talk about it all the time. But I, too, am constantly seeing in the mainstream press is this or that study saying about connecting Vitamin D levels to COVID outcomes. So when I finally saw this report that pulled it all together, and this rather breathtaking chart, I thought, okay, I need to take a moment to - and the competition for today's Picture of the Week was the BootHole logo, which is just wonderful. But we've got that down at the end.

Leo: Is it a boot with a hole? Okay, we'll get to that.

Steve: A boot with a big hole, yes. And a worm.

Leo: Which Vitamin D do you take? Vitamin D3?

Steve: Yes. You want D3. You know, Doctor's Best, NOW...

Leo: I take NOW, yeah.

Steve: Anybody who - yeah, in fact NOW is actually what I take. It's a little bottle because these things...

Leo: Oh, they're tiny little pills.

Steve: Even for people, yes, even for people who are not pill takers, this will not be a problem. It's a little tiny bit of a hormone, because it's not actually a vitamin, that is very diluted in olive oil because it is fat-soluble. You do not want to take more than 5,000 IU a day unless you're having your blood watched by a doctor. Back when we talked about it, Vitamin D wasn't even being tested. And now...

Leo: Yeah, my doc tests it.

Steve: ...because the mainstream medical community is beginning to catch on, they're actually taking a look at it.

Leo: Yeah, my doc tests it. And it's four cents a pill, the one I looked at.

Steve: Exactly.

Leo: It's not expensive.

Steve: I mean, exactly.

Leo: Okay.

Steve: So, yes, yeah. I mean, if you wanted to do one thing for the health of yourself and your family, I would say Vitamin D at a useful level. Oh, and really young kids ought to probably take less, maybe 2,000 IU.

Leo: Because it's the old folks who don't get any sun and don't convert it very well with their skin. Kids can go out and get some sun, and they'll make their own.

Steve: Yes, yes. Okay. So no big news on the Firefox front. The biggest new feature is a credential export which was added to Firefox's built-in Lockwise password manager. This exports the Firefox database into a CSV formatted text file, which you could then drop into a spreadsheet or import it into some other password manager, which is sort of why they say they did this. And of course it goes without saying that while exported in text form, it's readily discoverable by anyone or anything scanning your machine. So if you were to do that, storing it in a password-protected 7-Zip archive, which I think is probably the best of the free Zip things, 7-Zip is very popular.

Leo: And that's good protection?

Steve: Yes.

Leo: Because for a while the password-protected Zip files were not so strong. But this is good now?

Steve: Yes, they were a joke. Yeah, 7-Zip did it right. They derive a 256-bit AES key from a password-based key derivation function which uses a high iteration count for brute forcing delay, and we're going to be talking about brute forcing here before long because it's another mistake that Zoom made, to run an SHA-256 hash. So you do need to pick a good password. But if you do that with 7-Zip and its encryption, you should be safe to keep your passwords exported in that encrypted form. And I'll just note that I'm still using LastPass under Firefox because I also use it under Chrome and under Edge and under Safari and across all platforms. So, you know, it's worth noting that Firefox has a built-in password manager, but maybe as a backup it would be useful. So anyway, that's the news on Firefox 79. Not much there.

So I heard you talking about it on MacBreak, and so I just wanted to touch on this. We have learned more about who's behind, who's believed to be behind the Twitter hack. And, you know, not some powerful state-sponsored cybercrime gang; just a, we believe, a 17-year-old kid. His name is all over the tech press. I heard you not wanting to say it on MacBreak, Leo, but I do have it in the show notes.

Leo: It wouldn't be hard for somebody to find it. I mean, you know, I come from the school of journalism where you don't say the names of minors who are accused of crimes. But apparently nobody else does that.

Steve: Yeah. The local Florida news channel WFLA...

Leo: They doxed him right away.

Steve: Yeah. They outed him as Graham Clark from Tampa Bay, Florida.

Leo: They also published a suitably creepy picture of him.

Steve: I know. In fact, before I reduced it in size, and I actually had in the show notes, he looks a little bit like Spock.

Leo: Oh, he does.

Steve: Because he's got kind of a pointed ear. And he is a little bit creepy. And it's interesting, too, that his NIC is Kirk. So, you know?

Leo: Oh.

Steve: Maybe he does have pointy ears.

Leo: Yeah, yeah.

Steve: So people thought, you look a little Spock-like. Anyway, the sad thing is this guy's life is now seriously spocked-up.

Leo: Yeah, yeah.

Steve: He's been charged with 30 felonies relating to computer communications and organized fraud for scamming hundreds of people using compromised accounts. According to a press release from Hillsborough State Attorney Andrew Warren's office, this guy, Graham Clark we believe, now faces 30 felony charges. So we have one count of organized fraud involving more than $50,000; 17 counts of communications fraud of over $300; one count of fraudulent use of personal information in amount over $100,000 or 30 or more victims; 10 counts of fraudulent use of personal information; and one count of access to computer or electronic devices without authority and scheming to defraud.

So in total, 30 counts of felony charges. All of those are felonies. So, I mean, I do feel like, unfortunately, there's sort of a bit of overreaction. I mean, I get it that this was not good, and certainly the law enforcement wants to send a message, like don't do this, even if you can.

The initial announcement didn't indicate whether Clark had any partners in crime. But a few hours after the press conference announcement, the world learned that the U.S. DOJ had also filed charges against two other suspects believed to have helped Clark in this hack. The first of those was identified as Mason Sheppard, who's known as "Chaewon," 19 years old, living in Bognor Regis in the U.K. And the other is identified as Nima Fazeli, also known as "Rolex," a 22 year old residing in Orlando, Florida.

U.S. Attorney Anderson said: "There is a false belief within the criminal hacker community that attacks like the Twitter hack can be perpetrated anonymously and without consequence. Today's charging announcement demonstrates" - thus I think an example is being made - "that the elation of nefarious hacking into a secure environment for fun or profit will be short-lived. Criminal conduct over the Internet may feel stealthy to the people who perpetrate it, but there is nothing stealthy about it. In particular," he said, "I want to say to would-be offenders, break the law, and we will find you."

Leo: Oh, please. Exactly the kind of thing hackers go, hah. Hah, hah.

Steve: Uh-huh, yeah, thumb their nose.

Leo: Yeah. That's going to really scare me. I remember when I was a teenager.

Steve: Yeah. And in fact, Leo, this did - you know, I was always a good kid. But oh, you know, to be 17 and have this crazy...

Leo: Yeah, you do dumb things.

Steve: ...network in front of me, yeah.

Leo: Yeah, yeah.

Steve: So Twitter...

Leo: He was probably fairly clever because, well, go ahead. Because the way he did it was kind of interesting.

Steve: Yeah. So for their part, Twitter disclosed a bit more about the nature of the attacks. They said that the phone-based social engineering attack allowed the attackers to obtain the credentials of a limited set of employees which then made it possible to gain access to Twitter's internal network and support tools. Although not all of those employees who were initially targeted had permissions to use account management tools, the attackers, apparently actually just Graham, was able to use their credentials to then access Twitter's internal systems and gain information about Twitter's processes. That expanded knowledge then enabled the attackers to target additional employees who did have access to Twitter's privileged account support tools.

Reuters also had reported something that I had not seen elsewhere, which was that, as of earlier this year, more than a thousand Twitter employees and contractors had access to Twitter's internal tools and could change user account settings and hand control over to others. A thousand. And this was according to two former Twitter employees. Well, as we know, such widespread access makes it difficult, if not impossible, to defend against the sort of hacking that occurred.

So I did see some Disqus conversations. I'm sorry, Discord conversations. I read everything I could find, and ZDNet provided the most detail about the hack, including those, as I said, some Discord chat logs where Graham is seen soliciting the participation of the other two. He claims to work for Twitter and then offers to prove it by modifying their Twitter accounts. He also provided his bitcoin address. And Leo, I heard you mention on MacBreak Weekly the rather head-slapping fact that he provided his driver's license.

Leo: You have to at Coinbase to set up an account.

Steve: Yes. And so he sold them access to some high-value Twitter accounts such as @xx, @dark, @vampire, and @drug. Anyway, the link with all the details is in the show notes, for anyone who's interested.

So my take on this is that it's another example of what you might call "managerial inertia." And it was kind of natural. Let's remember that when Twitter was born, it wasn't initially taken very seriously. You know, it had that ridiculously limited text-only, patently insecure messaging of 140 characters max. I remember thinking, wait, 140 characters? That's it?

Leo: Yes. Those were the days.

Steve: But obviously over time Twitter's importance has grown dramatically. As we know, heads of industry and state use Twitter to reach their followers, including of course our U.S. President, who uses it to directly reach each one of his 84.5 million followers, multiple times per day. And more than likely, Twitter also didn't take itself very seriously at the start. You know. And as we've noted, there never really was any clear plan for how this free service was supposed to make any money. But over time, and very gradually, that changed.

So my point is, Twitter's importance doubtless crept up on it. Over the course of many years, Twitter slowly grew into a truly important global communications facility. As we know, it didn't start out as one, and it clearly is one today. That didn't happen all at once.

So I think this security breach is mostly a consequence of Twitter doing things the way it always had. And of any change to the status quo, you know, that occurred, the management just lagged behind. So my take on this is that this ultra high-profile security breach was probably the best thing that could have happened to Twitter. It did not actually result in a huge amount of damage. It has ruined a couple kids' lives, unfortunately. It'll be interesting to see what the sentencing is once this works its way through the courts.

But it was obviously, you know, if as Reuters reported at the beginning of the year, there were a thousand people who could do this, both inside and outside of Twitter, then this is a very much-needed wakeup call which was delivered probably in the nick of time. You know, we've got a very high-profile election coming up here in three months. We need the Internet to not betray the interests of the U.S. electorate. So I'm glad this happened, frankly, because it's clear that Twitter needs to get their act together, that they haven't been taking themselves as seriously as they need to. And again it's, you know, too bad for these young people.

The Garmin hack. Lawrence Abrams, BleepingComputer, as we know, has always had a strong interest in ransomware. So I'm not surprised that his coverage of the Garmin ransomware attack was the most detailed of any I've seen, nor that he's had access to some Garmin insiders who have reached out to provide him some extra tasty bits. Among other things, an employee inside Garmin informed him that the initial ransom demand was for $10 million.

Leo: Whoa.

Steve: We don't know - yeah.

Leo: Holy moly.

Steve: $10 million.

Leo: Okay.

Steve: We don't know what ransom was finally paid, but it seems more certain than ever that Garmin did pay up. Lawrence wrote: "After a four-day outage, Garmin suddenly announced that they were starting to restore services, and it made us suspect that they paid the ransom to receive a decryptor."

Then last Saturday Lawrence posted: "Today, BleepingComputer gained access to an executable created by the Garmin IT department to decrypt a workstation and then install a variety of security software on the machine. Since WastedLocker" - that's the ransomware - "is an enterprise-targeting ransomware with no known weaknesses in their encryption algorithm, a decryptor cannot be made for free." And remember that BleepingComputer has been sort of a focal point for the less-than-well-designed ransomware, where mistakes were found in the encryption which allowed for the creation of a no-charge decryptor, and those have been organized and can be found through BleepingComputer.

So, he said: "To obtain a working decryption key, Garmin must have paid the ransom to the attackers." Then this is where he said: "It's not known how much was paid; but as previously stated, an employee had told BleepingComputer that the original ransom demand was for $10 million. When extracted, this restoration package" - this is the one that they received a copy of that had been prepared by Garmin's IT department - "this restoration package includes various security software installers, a decryption key, a WastedLocker decryptor, and a script to run them all. When executed, the restoration package decrypts the computer and then preps the machine with security software.

"Garmin's script contains a timestamp of July 25, 2020, which indicates that the ransom was paid either on the 24th or 25th. Using the sample of WastedLocker from the Garmin attack" - that is, the actual ransomware from the Garmin attack - "BleepingComputer encrypted a virtual machine and tested the decryptor to see if it would decrypt our files." He said: "In our test, the decryptor had no problems decrypting our files."

So what was interesting was that the package received by BleepingComputer included references to both the cybersecurity firm Emsisoft, E-M-S-I-S-O-F-T, Emsisoft, and the ransomware negotiation service Coveware. When Bleeping Computer subsequently reached out to Coveware, they were told that they do not comment on any ransomware incidents reported in the media. And similarly, Emsisoft told BleepingComputer that they could not comment on any cases; that they create decryption tools and are not involved in ransom payments. Brett Callow, a threat analyst at Emsisoft, said: "I cannot comment on specific cases. But generally speaking, Emsisoft has no involvement whatsoever in negotiating or transacting ransom payments. We simply create decryption tools."

Okay. Now, that's interesting news. So it might seem odd for a reputable security firm such as Emsisoft to have anything to do with ransomware. But they have an interesting angle. As we know, the decryption side of the ransomware mess sometimes receives much less attention from the bad guys who need to create the decryptor than the encryption side. Consequently, the decryptors have tended historically to be buggy, to crash, or to for some reason fail to fully undo the damage that they had originally done, despite having received a valid key.

So that's where Emsisoft comes in. They reverse engineer questionable ransomware decryptors, for which the decryption key is known, to create a more robust and reliable decryptor for a victim's systems. Emsisoft's ransomware recovery services page states: "If the ransom has been paid, but the attacker-provided decryptor is slow or faulty, we can extract the decryption code and create a custom-built solution that decrypts up to 50% faster with less risk of data damage or loss." So this also explains why the decryption package Garmin finally used also contained legitimate security software. That extra security software, along with an improved decryptor, may have been provided by Emsisoft, or may have been put together by Garmin's IT.

And of course, as we mentioned last week, now that Evil Corp has been attributed as the creator of Wasted Locker and has been placed on the U.S. sanctions list for using Dridex to cause more than $100 million in financial damages, paying this ransom could lead to hefty fines from the government. So due to these sanctions, sources familiar with Coveware have told BleepingComputer that the negotiation company has placed WastedLocker on their own restricted list starting in early July and will not be handling negotiations for related attacks. So it does look like Garmin paid a ransom.

Leo: That's too bad. $10 million.

Steve: $10 million. Yeah. That was the initial demand.

Leo: Correct me if I'm wrong. I understand how hard it is to remediate after a ransomware attack. But if you have a backup of your data, there would be no reason to pay the ransom; right?

Steve: Right.

Leo: This is an implication that Garmin didn't have a copy of its data?

Steve: Well, I've seen reports. There was another firm - I can't remember. It was also in the news last week. They had 30,000 workstations encrypted. That demand started also at 10 million. They ended up settling, if you can call it that, to 4.5. So the negotiation through an intermediary came out at 4.5. I don't remember now the name of the company.

Leo: Because even if you did this, maybe, oh, it's 3,000 workstations, it'd be cheaper to pay and have them decrypted and then just working again. But would you ever trust that station? You wouldn't.

Steve: That's exactly the problem.

Leo: You're still going to have to wipe it and reinstall everything, no matter what you pay.

Steve: Yup.

Leo: I just don't get it. I'm missing something.

Steve: Well, my feeling is, I mean, certainly anybody now would have protection against, you know, like their main corporate databases would be secure. But it might just be that they're not doing nightly backups, like nightly images or incrementals of every single workstation in the organization. And so, you know...

Leo: Right. Well, they should.

Steve: Exactly.

Leo: You should have cold backups; right? You don't want hot backups because those could back up the encryption.

Steve: Right.

Leo: But this is well-known technology. This is not difficult to do.

Steve: True. I just think it's a matter of logistics. Like a large company sort of, you know, the IT department is busy running around remediating all manner of individual things. And it's like, okay, that's on our to-do list. Well, you know, I hope the industry is changing the priorities of these things because they really do need to get done.

Leo: Well, especially now. I mean, it's so obvious this is going to be a big business issue.

Steve: It is becoming big business.

Leo: Yeah.

Steve: So the Tor project has recently been in a bit of back-and-forth with a security researcher by the name of Dr. Neal Krawetz. Neal obtained his Ph.D. in Computer Science from Texas A&M and his bachelor's from UC Santa Cruz. He has a long history of finding and reporting problems with the Tor Network, and he operates multiple Tor nodes himself. From looking over the history of this, he appears to have long been a bit of a thorn in the side of the Tor engineers, and frankly not all of his concerns over Tor's privacy guarantees appear to warrant undue concern.

For example, he wrote at one point over, well, in fact, in ramping up to his decision finally to disclose something without Tor's permission. He said: "Over three years ago, I tried to report a vulnerability in the Tor Browser to the Tor Project. The bug is simple enough. Using JavaScript, you can identify the scrollbar width. Each operating system has a different default scrollbar size, so an attacker can identify the underlying operating system. This is a distinct attribute that can be used to help uniquely track Tor users." And he says in parens: "(Many users think that Tor makes them anonymous. But Tor users can be tracked online. They are not anonymous.)"

So anyway, okay. In three years, despite trying, he had not managed to get this fixed. During that time the Tor Project joined HackerOne, you know, the firm that we've talked about often, for creating bug bounties, and officially credited him for the discovery of this problem, the fact that JavaScript running in a browser could determine the width of the scroll bar. Okay. They credit him with the discovery of that, and I don't know whether he received any monetary payment. But the resolution of this was deferred from Tor to Mozilla, since after all the Tor browser is based on Firefox, and that's Mozilla's baby.

After some length of time doing nothing with this, it was dropped, just the guy to whom it was assigned deassigned himself from it, and that upset the good doctor. And of course we've seen instances of this before where a security researcher finds a problem that he or she believes to be highly critical and that needs everyone's attention right away, you know, if not yesterday. But for whatever reason they don't obtain the satisfaction that they're looking for from the affected parties. They feel unappreciated for their efforts, stiffed and ignored.

So then what comes next? In the case of this Dr. Neal Krawetz, under the subheading "Dropping Zero-Days," he now explains on his most recent blog posting, and in fact he uses the definition that I don't agree with. He starts off: "A 'zero-day' is any exploit that has no known patch or widespread solution." And of course, as we know, I disagree with that. I think that it's just an unknown vulnerability unless and until it is found to be exploited.

Anyway, he continues: "A zero-day doesn't need to be unique or novel; it just needs to have no solution." He says: "I'm currently sitting on dozens of zero-days for the Tor browser and Tor network." He says: "Since the Tor Project does not respond to security vulnerabilities" - and in fact they do - "I'm just going to start making them public. While I found each of these on my own, I know that I'm not the first person to find many of them." Well, okay. He infers that, I suppose.

So here we have the unfortunate phenomenon of the security researcher whose original white hat begins to dim as his or her work doesn't receive the attention they believe it deserves. So he continues: "The scrollbar profiling vulnerability is an example of a zero-day in the Tor browser." And I'll just say as an aside, we can hope that all of these other dozens of zero-days are of similar impact. He says: "But there are also zero-days for the Tor network. One zero-day for the Tor network was reported by me to the Tor Project on December 27th, 2017," he says in parens, "(about 2.5 years ago). The Tor Project closed it out as a known issue, won't fix, and informative."

He says: "Let's start with a basic premise. Let's say you're like some of my clients. You're a big corporation with an explicit 'No Tor on the corporate network' rule. This is usually done to mitigate the risks from malware. For example, most corporations have a scanning proxy for Internet traffic that tries to flag and stop malware before it gets downloaded to a computer in the company. Since Tor prevents the proxy from decoding network traffic and detecting malware, Tor is not permitted. Similarly, Tor is often used for illegal activities." And he cites child porn, drugs, et cetera.

"Blocking Tor reduces the risk from employees using Tor for illegal purposes. Although denying Tor can also mitigate the risk from corporate espionage, that's usually a lesser risk than malware infections and legal concerns." And he says: "Keep in mind these same block and filtering requirements apply to nation-states like China and Syria that want to control and censor all network traffic." He says: "But I'm going to focus on the corporate environment."

"It's one thing to have a written policy that says 'Don't use Tor.' However, it's much better to have a technical solution that enforces the policy. So how do you stop Tor users from connecting to the Tor network? The easy way, he says, is to download the list of Tor relays. A network admin can add a fire rule blocking access to each Tor node."

Then he says: "Zero-day number one," this apparently the beginning of dozens. "Blocking Tor connections the smart way." He says: "There are two problems with the 'block them all' approach. First, there are thousands of Tor nodes. Checking each network connection against every possible Tor node takes time. This is fine if you have a slow network or low traffic volume, but it doesn't scale well for high-volume networks. Second, the list of nodes changes often. This creates a race condition, where there may be many new Tor nodes that are seen by Tor users but aren't blocked by your network block list yet."

He says: "However, what if there was a distinct packet signature provided by every Tor node that can be used to detect a Tor network connection? Then you could set the filter to look for the signature and block all Tor connections. As it turns out, this packet signature is not theoretical." And then in his blog posting he goes on to describe in great detail Tor's TLS handshake and the unique properties which he found and told Tor about 2.5 years ago of the TLS certificates which Tor node servers generate on the fly.

Then he says, finally, he says: "Validating the vulnerability. Back in 2017, I used a scanner and Shodan to search for TLS certificates. In theory, it's possible for there to be some server with a server-side TLS certificate that matches this signature, but that is not a Tor node. In practice, every match found was a Tor node." He said: "I even found servers running the Tor daemon and with open onion routing ports that were not in the list of known Tor nodes."

He says: "Some were non-public bridges. Others were private Tor nodes. Similarly, I scanned every known Tor node. Each matched this Tor-specific signature profile. That makes the detection 100% accurate, no false positives, no false negatives." He says: "Although now that I've made this public, someone could intentionally generate false-positive or false-negative certificates. The false-positives are relatively easy to construct. The false-negatives will require editing the Tor daemon's source code.

"While a scanner could be used to identify and document every Tor server," he says, "corporations don't need to do that. Corporations already use stateful packet inspection on their network perimeters to scan for potential malware. With a single rule, they can also check every new connection for this Tor signature. Without using large lists of network addresses, you can spot every connection to a Tor node and shut it down" - that is, shut the connection down - "before the session layer (TLS) finishes initializing, and before any data is transferred out of the network."

So he then explains that he reported this discovery of a simple way of detecting and thus blocking all Tor traffic. He said: "I reported this simple way to detect Tor traffic to the Tor Project on" - as we said before - "27th of December, 2019, HackerOne bug #300826." Meaning that HackerOne has acknowledged it, and presumably he's been paid a bounty for it. He says: "The response I got back was disappointing." Or in fact maybe the response means he didn't get paid.

Tor replied: "Hello, and thanks for reporting this issue! This is a known issue affecting public bridges, the ones distributed via bridgedb. See ticket #7349 for more details. This issue does not affect private bridges, the ones that are distributed in a peer-to-peer ad hoc way. As indicated in the ticket, to fix this problem, we are aiming to make it possible to shutdown the ORPort" - the onion routing port - "of Tor relays. In our opinion, we should not try to imitate normal SSL certs because that's a fight we can't win. They will always look different or have distinguishers, as has been the case in the pluggable transport arms race. Unfortunately, ticket #7349 is not straightforward to implement and has various engineering complexities. Please see the ticket for more information. Due to the issue being known and planned to be fixed, I'm marking this issue as 'Informative.'"

So needless to say, the doctor was displeased, and his blog posting itemized his disagreements with this decision. You have to take my word for it. I won't go over them. Then he concludes with some commentary on bug bounties and a promise for more zero-days, which I think are simply presently known vulnerabilities. He wrote: "More soon. If you have ever worked with bug bounties, then you are certain to recognize the name Katie Moussouris." And of course we've talked about her in the past.

"She created," he says, "the first bug bounty programs at Microsoft and the Department of Defense. She was the Chief Policy Officer at HackerOne, the bug bounty service, and she spearheaded NTIA's Awareness and Adoption Group's effort to standardize vulnerability disclosure and reporting." And he says, parens: "(Full disclosure: I was part of the same NTIA working group for a year.)" He said: "I found Katie to be a positive and upbeat person. She is very sharp, fair-minded, and realistic."

So then he said: "Earlier this month, Katie was interviewed by the Vergecast podcast." He said: "I had expected her to praise the benefits of vulnerability disclosure and bug bounty programs. However, she surprised me. She has become disenchanted by how corporations are using bug bounties. She noted that corporate bug bounties have mostly been failures. Companies often prefer to outsource liability rather than solve problems.

And they view the bug bounties as a way to pay for the bug and keep it quiet rather than fix the issue.

"Every problem that Katie brought up about the vulnerability disclosure process echoed my experience with the Tor Project. The Tor Project made it hard to report vulnerabilities. They failed to fix vulnerabilities. They marked issues as 'resolved' when they were never fixed. They outsourced simple issues, like passing a simple scrollbar issue upstream to Firefox where it is never fixed. And they make excuses for not addressing serious security issues." And we'll just note what he considers to be serious security issues.

"During the interview, she mentioned that researchers and people reporting vulnerabilities only have a few options: try to report it, sell it, or go public." He said: "I've tried reporting and repeatedly failed. I've sold working exploits, but I also know that they can be used against me and my systems if the core issues are not fixed. And even the people who buy exploits from me would rather have these issues fixed. That leaves public disclosure." He says: "In future blog posts, I will be disclosing more Tor zero-day vulnerabilities. Most, but probably not all, are already known to the Tor Project. I have a list of vulnerabilities ready to drop. And for the Tor fan boys who think 'use bridges' will get around this certificate profiling exploit, don't worry. I'll burn bridges next."

So anyway, I thought this story was interesting and worth covering and sharing with our listeners because it illuminates another facet of this weird security industry that we spend time looking at every week. It's certainly the case that a vulnerability hunter lacks the ability to force their discoveries to be fixed. But I think that forcing discoveries to be fixed is probably the wrong goal. If after having been informed of it, an organization should choose not to repair a defect in their system for whatever reason, isn't that entirely their business?

I mean, I understand the ego involvement and the temptation to force the issue. The security researcher is in possession of knowledge; the public is not. But attempting to publicly shame an organization into bending to the attacker's will feels wrong, especially when that public shaming must, in order to be effective, inherently put other users of the organization's systems at some form of increased risk. I mean, that's the nature of the shame. So it's clear how a formal bug bounty program such as HackerOne could be abused by an organization to purchase and then sit on their bugs. But again, isn't that exactly the right that they have purchased? That's part of the bargain. The bug hunter agrees not to disclose, and in return receives payment, both for the documentation of the discovered problem and for their continued silence about it. What happens after that is no longer the hacker's business. That information has been sold.

So anyway, I thought it was interesting. I have not made time to listen to Katie's conversation with the Verge. But if I find time and can find it, I think I will because I'm kind of curious to hear what someone who was a big proponent of this economic model for monetizing the work of investigators and, I mean, arguably resulting, as we've talked often about, you know, much heightened security overall for the industry. I guess no system works perfectly all the time. And I heard you sort of in some agreement in the background, Leo.

So I think that the core lesson of this next story is, in this day and age, in 2020 and beyond, it's truly necessary to do everything right. Which brings us to the latest Zoom mistake. And this one's not a bug. It's a design mistake. Back in April, in response to the flurry of interest in Zoom, both by the overworld, who wanted to use it, and the underworld, who wanted to abuse it, Zoom bombing, as we know, became a thing. Which caused us to title an episode "Zoom Go Boom."

The trouble was that Zoom meetings were not required originally to have any kind of password protection. Just the meeting code was sufficient to allow otherwise uninvited visitors to break into and disturb Zoom conferences of all kinds. Zoom quickly responded by adding a six-digit PIN to protect entry into all recurring Zoom meetings. And as we know, a six-digit PIN can provide some useful security. After all, that's what our authenticators all use. But it must be deployed with some care because it does not by itself provide much entropy.

And sure enough, Tom Anthony, the VP in charge of product at SearchPilot in the U.K., discovered that Zoom's implementation of six-digit PINs had made a classic rookie mistake. And frankly, in this day and age, it's kind of unforgiveable. But I'm sure they were in a hurry to close down the Zoom bombing problem.

Okay. What Tom discovered, somewhat to his amazement, was that Zoom had failed to implement any sort of rate limiting to prevent high-speed brute-force guessing of these comparatively short six-digit numeric PINs. Like I said, a rookie mistake. As Tom wrote in his write-up, this enabled, he said, "an attacker to attempt all one million possible PINs in a matter of minutes and gain access to other people's private Zoom meetings."

So as we know, in the absence of any checks for repeated incorrect password attempts, which would lock out an IP, nor any rate limiting for mistakes, and really, when you think about it, correctly entering a six-digit PIN? That's just not difficult to get right, if you know what it is. So it would make sense for the entry system to be highly intolerant of what is clearly guessing. So none of that was present. So an attacker could leverage Zoom's web client. And remember, it's a simple URL, https://zoom.us/j/, then the meeting ID, to continuously send these HTTP requests until one million combinations have been tried.

And he noted, with improved threading and distributing, the guessing clients across maybe four or five cloud servers, the entire six-digit, one million possible PIN space could be checked within a few minutes. He responsibly reported this glaring oversight to Zoom on April 1st, along with a Python-based proof-of-concept script. The next day, Zoom took their web client offline, since that was the largest and most glaring exploit vector. And then a week later they fixed the flaw permanently and correctly.

So it's obviously good that this was caught early and fixed quickly. But the lesson here is that this should never have happened in the first place. The problem we have today is that truly important security pieces are still being implemented in an ad hoc fashion. You know? Like everyone is rolling their own, every time they need a solution. They're still needing to reinvent the same wheel over and over again. And this approach invites mistakes. Even if people knew to do it right, I mean, it's unbelievable that somebody could, in this day and age, implement a six-digit PIN where no measure was taken to prevent brute forcing. But that's what Zoom put online.

Today, every developer rolls their own web UI to suit their particular needs, so every one of them is different. Everyone of them handles login a little differently. There's no uniformity about passwords. Can I use a special character? How long can it be? What if I forget it? Everyone handles recovery slightly differently. What we have today is a mess. And not only does user convenience suffer, but so does security. So it's going to be interesting to see how this gets resolved, you know, downstream.

We all know that I designed something that attempted to unify this process, but it's something that needs to get adopted. A solution needs to get adopted industry-wide. And it's going to have to be a solution that broadly solves the problems and that doesn't require everybody to reroll their own solution, or we're not going to get ahead of this.

The good news is, if this was April 1st, this was probably before the heavyweights got involved, and so this was still hopefully the original team at Zoom who said, well, you know, let's just create a quick solution. We'll see.

Another SHA-1 deprecation. In their posting titled "SHA-1 Windows content to be retired August 3rd, 2020" - in other words, since today's the 4th, that was yesterday - last week Microsoft made the following announcement. They said: "To support evolving industry security standards and continue to keep you protected and productive" - pardon me, that's assuming Windows 10 will boot, or that you could print after it does - "Microsoft will retire content that is Windows-signed for Secure Hash Algorithm 1 (SHA-1) from the Microsoft Download Center on August 3rd, 2020."

They said: "This is the next step in our continued efforts to adopt Secure Hash Algorithm 2 (SHA-2), which better meets modern security requirements and offers added protections from common attack vectors. SHA-1," they wrote, "is a legacy cryptographic hash that many in the security community believe is no longer secure. Using the SHA-1 hashing algorithm in digital certificates could allow an attacker to spoof content, perform phishing attacks, or perform man-in-the-middle attacks. Microsoft no longer uses SHA-1 to authenticate Windows operating system updates due to security concerns associated with the algorithm, and has provided the appropriate updates to move customers to SHA-2 as previously announced. Accordingly, beginning in August 2019" - so a year ago - "devices without SHA-2 support have not received Windows updates. If you are still reliant on SHA-1, we recommend that you move to a currently supported version of Windows and to stronger alternatives such as SHA-2."

The only consequence I can see this having would be those among us who have some reason to set up and update older versions of Windows. For example, I was just recently testing SpinRite's forthcoming USB prep technology - which I, as we know, packaged as "InitDisk" - under Windows XP because it still needs to run there; and Windows 7, which continues to valiantly hold onto to a bit more than a quarter of the desktop market share. Last time I looked it's at 26.72%. It originally shipped without support for SHA-256, as did Windows XP.

So I wonder whether there will be a bit of a Catch-22 because there are Windows updates that are required to add SHA-256 support to Windows XP and Windows 7. So they must still be signed with SHA-1 in order to allow SHA-256 support to be bootstrapped onto those earlier operating systems. Fortunately, I long ago created kits of all the required files, so I'm okay. And I imagine that that's also true for most others who have a similar interest in archaeology. But I did, when I thought, whoa, SHA-1's going away? Uh-oh. Good thing I've got RAID-based archives in multiple places of those particular files that allow the older OSes to understand signing with SHA-256. And, yeah, archaeology.

Speaking of which, QNAP and QSnatch. QNAP Network Attached Storage devices, well, they've been giving their owners and the security industry some serious problems for nearly a year, though the troubles appear to date as far back as 2014, so six years. We've touched on this before, but it's worthy of a brief refresher because there's been a recent escalation in the breadth and depth of the attacks against still unpatched QNAS devices. Last week the cybersecurity agencies in both the U.S. and the U.K. issued a joint advisory about a massive ongoing malware threat which is infecting the NAS appliances of QNAP, a Taiwanese-based company.

The malware goes by the name QSnatch, or for some reason Derek. I'm reminded of that - have you seen the auto insurance commercial where the gal is talking to Bigfoot, and he's lamenting the fact that no one really cares about him anymore.

Leo: Yes, his name is Derek, yes. Bigfoot? But my name's Derek.

Steve: So QSnatch, also known as Derek, is a credential and data-stealing malware which has compromised more than 62,000 devices since reports began last October. There's a high degree of infection in North America and Western Europe.

Leo: Oh, yeah. We love them. They're great. But maybe not so great as they used to be. But yeah.

Steve: Yeah. So it's probably for the best that even today the exact infection vector is not publicly known. It has not been disclosed. But the U.S. Cybersecurity and Infrastructure Security Agency (CISA) and the U.K.'s National Cyber Security Centre (NCSC) wrote in their joint alert that "All QNAP NAS devices are potentially vulnerable to QSnatch malware if not updated with the latest security fixes." And also that "Once a device has been infected, attackers can prevent admins from successfully running firmware updates." So talk about a Catch-22.

QSnatch has an assortment of features which are implemented as modules. Therefore the problems include, but are not necessarily limited to, a password logger which installs a fake device admin login page to spoof people into entering their credentials, which it then grabs; a more generic credential scraper; an SSH backdoor enabling the attackers to run arbitrary code on the infected devices; a web shell to provide malware operators with remote access to compromised NASes; and a data theft module which steals a predefined list of files, including logs and system configuration, and sends them in encrypted form to attacker-controlled servers. So in other words, you don't want this to be running in your network attached storage. But getting rid of it is tricky. It requires multiple reboots, full firmware downloads, some sort of a malware-scraping utility, and more.

I've got a link to the QNAP advisory. At QNAP they say: "QSnatch collects confidential information from infected devices, such as login credentials and system configuration. Due to these data breach concerns, QNAP devices that have been infected may still be vulnerable to reinfection after removing the malware." In other words, don't just flush the malware, but then retain the use of your favorite passwords in the device for the sake, for example, of connectivity with other applications or users who may have and be using the previous credentials.

You should assume that a complete compromise of all the secrets in the device has already taken place, including all accounts on it. And be also wary of any add-on software. You know how lots of these network attached storage devices now you're able to install all sorts of other goodies. Don't have any there that you're not using, and get rid of any that you're unsure about. Anyway, it's a mess. If you own a QNAP NAS, it's worth some time to make sure that it's clean.

My discussion last week of the forthcoming mass storage benchmark, which will be another development spinoff on the ongoing work toward SpinRite 6.1, it generated a lot of interest and feedback from our listeners. Many people wanted to know where it was and how they could run it. So I need to quickly note that even it, the benchmark, is still in development and not yet ready for general use. And believe me when I say that at this point it would cause far more confusion, frustration, and questions than it would answer because it is just a development tool. But as soon as it's ready for general purpose use, I will make it easy to find, and I will formally invite all of our listeners to experiment with it.

I mentioned last week that I was going to add further granularity to the benchmark to look at the timing of the individual 32MB transfers which make up the larger 1GB benchmark. We did that, and the results were very interesting. We definitely found spots where drives, both spinning and solid state, but interestingly primarily solid state, were a great deal slower to respond. And in general we're seeing much more evidence that highly used regions of SSDs, typically at the front of the drive underneath the operating system, are consistently performing much more slowly, sometimes at as little as half the speed as compared to the unused areas.

We know that SSDs broadly employ two management schemes to compensate for the technology's inherent lack of write endurance. They perform wear leveling, which dynamically relocates the data from the more highly used silicon to the lesser used regions. And just as with hard drives, SSDs are also generously overprovisioned to allow regions that finally have been worn out to be replaced with fresh storage that had been set aside for that purpose.

So what we think we're seeing and that is being revealed by the benchmark could be the extra time being required for error correction, which would tend to be required to fix low bit count errors as memory is becoming fatigued. And we might also be seeing evidence of some overhead associated with the management of what eventually becomes physically fragmented solid-state storage. In any event, this doesn't appear to be something that there's much awareness of today, but this benchmark reveals it conclusively. And I imagine that our listeners are going to find this fascinating.

So stay tuned. I have a few more things to deal with, and then I'll be integrating this final AHCI driver part into the earlier IDE and compatibility and legacy mode drivers to produce a single result which should run on everybody's hardware, and then the fun will begin.

Leo: Okay, Steve. Shall I show the logo? Look at that. BootHole.

Steve: Yeah. You think it was, like, preexisting artwork?

Leo: That looks like clipart, yeah.

Steve: I guess maybe, yeah.

Leo: It kind of does. I don't know. What do you think? Maybe not the worm, but the boot, definitely. The worm looks like somebody drew it themselves, kind of. It's cute. It's very cute.

Steve: So early in the history of this podcast, well before modern secure booting was actually invented, we talked about how truly insidious rootkits could be. Remember those episodes, Leo? We had a lot of fun with that.

Leo: Oh, yeah.

Steve: Was it Sony that was hiding itself?

Leo: Sony, yeah.

Steve: Crazy.

Leo: They put DRM in a rootkit. That's a good idea.

Steve: Right. By hooking, and basically just to hide its own files. And in this case by hooking and subverting the operating system's own file system, a user could be looking right at a directory containing malicious malware and not see it. You do a directory listing, and the rootkit's API hooks would filter out any and all appearance of any files it didn't want you or your AV or anything else to see. You know, scanners wouldn't see it. You wouldn't. Nobody would see it. But the files would still be right there, like in front of you, unseen. So rootkits are a big problem when the goal is to have a truly trustworthy system.

And we've previously covered the concept of secure booting thoroughly over many previous podcasts, both in the context of securely booting a PC and also iOS that has a similar Secure Boot technology. The idea is the establishment of a chain of trust which is anchored by some root component that can be absolutely trusted, and which is then able to examine and verify the trustworthiness of each and every subsequent stage of the booting process. With secure booting enabled, the integrity of the resulting system is supposed to be assured and something that you can assume.

So when two researchers - Mickey, looks like Shkatov, and Jesse Michael, both at Eclypsium - announced their discovery of a vulnerability which they called BootHole in the GRUB2 bootloader used by most Linux systems, which can be used to gain arbitrary code execution during the boot process even with Secure Boot enabled, this understandably generated quite a stir within the security industry. This meant that attackers could exploit this vulnerability to break boot security and install persistent and stealthy bootkits - and bootkits are another name for rootkits, essentially - to provide near total control over the victim device. GRUB, G-R-U-B, stands for Grand Unified Bootloader.

I've got a link to their full description, a PDF link in the show notes. But in their disclosure of this, they give a nice summary. They wrote: "The vulnerability affects systems using Secure Boot, even if they are not using GRUB2. Almost all signed versions of GRUB2 are vulnerable" - and there's one that isn't, but otherwise they're all known to be vulnerable - "meaning virtually every Linux distribution is affected. In addition, GRUB2 supports other operating systems, kernels and hypervisors such as Xen.

"The problem also extends to any Windows device that uses Secure Boot with the standard Microsoft 3rd Party UEFI Certificate Authority. Thus the majority of laptops, desktops, servers, and workstations are affected, as well as network appliances and other special purpose equipment used in industrial, healthcare, financial, and other industries. This vulnerability makes these devices susceptible to attackers such as the threat actors recently discovered using malicious UEFI bootloaders." In other words - this is me speaking - the idea of corrupting a system's boot is not just theoretical, it is actively happening in the wild today.

So they continue: "Eclypsium has coordinated the responsible disclosure of this vulnerability with a variety of industry entities, including OS vendors, computer manufacturers, and cybersecurity emergency response teams. Mitigation will require new bootloaders to be signed and deployed, and vulnerable bootloaders should be revoked to prevent adversaries from using older, vulnerable versions in an attack." In other words, the bootloaders, the vulnerable bootloaders are currently signed and trusted by the root of trust in the UEFI on the motherboard. So this is an instance where revocation of some form is required. They finish: "This will be a long process and considerable time for organizations to complete patching." In other words, this is a big whoopsie.

However, nobody should go out and fix this right now. We'll get to why in a second, because it is actually causing way more problems than it is worth. I'm going to cut to the chase here because part two of this is what has happened since. In their disclosure document, they explain the problem. They said: "In the course of Eclypsium's analysis, we have identified" - guess what - "a buffer overflow vulnerability in the way GRUB2 parses content from the GRUB2 config file named grub.cfg." They said: "Of note, the GRUB2 config file is a text file and typically is not signed like other files and executables." And I'll note that that one GRUB2 that I mentioned that is not vulnerable to this is not vulnerable because it requires the grub.cfg to be signed.

So they said: "This vulnerability" - this buffer overflow in the parsing of grub.cfg - "enables arbitrary code execution within GRUB2 and thus control over the booting of the operating system. As a result, an attacker could modify the contents of the GRUB2 configuration file to ensure that attack code is run before the operating system is loaded. In this way, attackers gain persistence on the device.

"Such an attack would require an attacker to have elevated privileges. However, it would provide the attacker with a powerful additional escalation of privilege and persistence on the device, even with Secure Boot enabled and properly performing signature verification on all loaded executables. One of the explicit design goals of Secure Boot is to prevent unauthorized code, even running with admin privileges, from gaining additional privileges and pre-OS persistence by disabling Secure Boot or otherwise modifying the boot chain.

"With the sole exception of one bootable tool vendor who added custom code to perform a signature verification of the grub.cfg config file in addition to the signature verification performed on the GRUB2 executable, all versions of GRUB2 that load commands from an external grub.cfg config file are vulnerable. As such, this will require the release of new installers and bootloaders for all versions of Linux." In other words, this is not a quick, easy, small fix.

"Vendors will need to release new versions of their bootloader shims to be signed by the Microsoft 3rd Party UEFI Certificate Authority. It is important to note that until all affected versions are added to the dbx revocation list" - and I'll explain that in a second - "an attacker would be able to use a vulnerable version of shim and GRUB2 to attack the system. This means that every device that trusts the Microsoft 3rd Party UEFI Certificate Authority will be vulnerable for that period of time."

Okay. So first, as regards the buffer overflow, UEFI does not employ address space layout randomization, data execution prevention, or any of the other common exploit mitigation protections that have fortunately become standard in our operating systems. This means that weaponizing this buffer overflow will be trivial for attackers who already have a foothold on the targeted computer to exploit the flaw.

Leo: By the way, underline that part, "who already have access to the attacked computer."

Steve: Yes, exactly. That's why nobody needs to actually huff and puff and run around in circles and worry about this.

Leo: Incidentally, I've never installed Linux without turning off Secure Boot because most Linuxes aren't signed.

Steve: Right. It gets in the way, exactly.

Leo: Yeah. So this would be for enterprise users who are using signed Linuxes and so forth.

Steve: Correct. Well, and for example, it is Red Hat Enterprise Linux that has been found to have a problem. We'll get there in a second. But I was going to say that GRUB2 is also open source, and that we already have full open documentation of the problem. So bad guys, you know, maybe there will be a bit of a race. But again, Leo, as you highlighted, this requires modifying a config file that is stored in the UEFI that no one can get to unless they have physical access to the system or already have elevated privileges.

Leo: Right.

Steve: So it makes a serious problem worse and persistent. And if you didn't know if you had this, if something got into your system and you had it, could arrange persistence that a reformat of your hard drive would not resolve.

Leo: That's the key. Yeah, that's good, yeah.

Steve: Yeah. So we talked about this before, but it's worth noting that, thankfully, the Secure Boot system was understood to require, from the beginning, some form of truly effective revocation mechanism. Not the mess that we have with our browser certificates. So every UEFI system which supports Secure Boot contains a pair of protected databases. The "Allow DB," which is just called "db," lists the approved components, and the "Disallow DB," which is called "dbx," contains a list of known vulnerable or malicious components including firmware, drivers, and bootloaders.

So what this means is that all previously vulnerable bootloading components, all of these GRUB, the various GRUB2s that are out there and are signed and are trusted and are now known to be exploitable, they all have to be added to the Disallow DB so that a bad guy who does get this kind of access can't swap out your good GRUB2 for one of these bad GRUB2s. So it's a big problem to remediate. And this has to happen to every single motherboard where trusted Secure Boot wants to be used.

But Leo, wait. There's more. And it's, like, oh my god more. In response to Eclypsium's initial vulnerability report, the GRUB2 code came under additional and, as it turns out, very much needed scrutiny. A distressing number of additional vulnerabilities were then discovered by the Canonical security team. CVE-2020, and I'll skip that preamble from now on, 14308 GRUB2: grub_malloc does not validate allocation size, allowing for arithmetic overflow and subsequent heap-based overflow. 14309 GRUB 2: Integer overflow in grub_squash_read_symlink may lead to heap-based overflow.

14310: Integer overflow read_section_from_string may lead to heap-based overflow. 14311: Integer overflow in grub_ext2_read_link leads to heap-based buffer overflow. 15705: Avoid loading unsigned kernels when GRUB is booted directly under Secure Boot without a shim. 15706 script: Avoid a use-after-free when redefining a function during execution. And, finally, 15707, what is it, the seventh? Yes, the seventh additional CVE: Integer overflow in initrd size handling.

So, yes, GRUB2 turns out to have had a lot of problems, and it finally came to the security industry's attention. And given the difficulty of this scale, this kind of ecosystem-wide update and revocation, there is a strong desire to avoid having to do this again in six months, so a large effort spanning multiple security teams at Oracle, Red Hat, Canonical, VMware, and Debian, using static analysis tools and manual code review, have identified and fixed dozens of additional vulnerabilities and dangerous operations throughout this GRUB2 codebase that do not yet have additional or have individual CVEs assigned. So yeah, Leo, you might as well have just turned off Secure Boot because I don't think it was really doing anything anyway.

Leo: That secure anyway, yeah.

Steve: So what needs to be done now to fully respond to the revelation after this flaw? Broadly, five things, and don't do them. Updates to GRUB2 to address the vulnerability. Don't do that. Linux distributions and other vendors using GRUB2 will need to update their installers, bootloaders, and shims. And actually they will need to reupdate them. New shims will need to be signed by the Microsoft 3rd Party UEFI Certificate Authority. Administrators of affected devices will need to update installed versions of operating systems in the field, as well as installer images, including disaster recovery media. And don't do that yet. Eventually, the UEFI revocation list (dbx) needs to be updated in the firmware of each affected system to prevent running any of the previously trusted, now known to be insanely vulnerable, code during boot.

We never talked about the need for shims, and I've used that term a couple of times. Open source projects and other third parties create a small app called a "shim." It works, well, it contains the vendor's certificate and code that verifies and runs the bootloader. The vendor's shim is verified using the Microsoft 3rd Party UEFI Certificate Authority, and then the shim loads and verifies the GRUB2 bootloader using the vendor certificate embedded inside the shim. In other words, it's a means of installing essentially a third-party certificate, which is then used by other projects like open source projects, which are signed by that so that they're able to participate in this whole Secure Boot project, as well.

While it's certainly true that Secure Boot should be made as secure as we can make it, some knowledgeable security industry insiders feel that way too much ado is being made of this whole thing. Set aside the fact that it was also badly broken initially. We'll get there in a second. But we know of HD Moore. He's the widely acknowledged expert in vulnerability exploitation who was the original developer of the Metasploit framework.

He told Ars Technica's Dan Goodin in an interview, he said: "I'd argue that Secure Boot is not the foundation of PC security today because it is rarely effective; and, by Eclypsium's own claim, it has been easy to bypass for over a year now, with no long-term fixes in sight. I'm not sure what the buffer overflow in GRUB2 is useful for, since there are other problems if the grub.cfg is unsigned. It may be useful as a malware vector; but even then there is no reason to exploit a buffer overflow when a custom grub.cfg file can be simply used to chain load the real operating system." He's saying, in other words, if you're going to change GRUB2, why bother with a buffer overflow? Just have it load something else of your choice first.

So, still, we want GRUB2 to be as secure as it can be, as you said, Leo, for enterprise environments. And there's an aspect of this that's reminiscent of Spectre and Meltdown, as I mentioned, where the cure is arguably worse than the problem because Red Hat's patch to GRUB2 and the kernel, once applied, is now rendering those systems completely unbootable. The issue has been confirmed to affect Red Hat Enterprise Linux 7.8 and 8.2, and may also affect 8.1 and 7.9. The derivative distribution CentOS is also affected.

Consequently, Red Hat is now advising users not to apply the GRUB2 security patches until these initial issues have been resolved. They say, if someone has installed the fix, do not reboot your system. Downgrade the affected patches. And if the patches were applied and the system reboot was attempted and failed, users should boot from an RHEL or CentOS DVD in its troubleshooting mode, set up the network, then back out to restore the system's original boot.

Additionally, although the problem was first reported in Red Hat Enterprise Linux, apparently related bug reports are now rolling in from other distributions from other families, as well. Ubuntu and Debian users are reporting systems which cannot boot after installing the GRUB2 updates, and Canonical has issued an advisory, including instructions for recovery on affected and no longer bootable systems.

So it's certainly good that the GRUB2 code got a clearly, very clearly much-needed close examination with many fixes. I mean, dozens. But this particular problem requires, as I said, the grub.cfg file to first be somehow maliciously altered. And doing that would require physical access to the system or elevated privileges. And at the moment, updating to fix this might render one's system completely unusable.

The obvious advice, since the sky is not actually falling, would be to wait a while until all the dust from this has settled, and the various kinks have been worked out of the process. Then have a leisurely update and know that a bunch of potentially exploitable flaws have been fixed. And that's a good thing. But it's nice to hear, Leo, that you're not running with Secure Boot turned on because of course that's also a problem for SpinRite that I will have to be dealing with at some point. If Secure Boot ever actually becomes an issue, you know, I will need to be able to get SpinRite to boot. Well, either to have a user briefly disable it, if it was enabled, but I'm also seeing the same thing. Nobody's running with it on. It's just kind of in the way. And we know now...

Leo: Windows users are. I mean, if you buy a Windows machine, and you don't do anything, it's going to have Secure Boot.

Steve: Right.

Leo: But no Linux user would because, well, not "no." I mean, I guess in enterprise there are signed versions of Linux. But why should I take my version of Linux to Microsoft to get it signed so I can - it doesn't make any sense. For a long time we thought Secure Boot was a conspiracy by Microsoft to damage Linux. We now know that's not true. But yeah, I just turn it off, usually. Most of the time it's easier to install Linux that way. And I'm glad to know it's not that secure.

Steve: Nothing to see. Move along.

Leo: Yeah. It's pretty funny.

Steve: Wow.

Leo: That's interesting that you can't - because you need to boot up clean to run SpinRite. So you would need a signed version of, well, you're not using FreeDOS anymore; right? You're going to just boot directly?

Steve: Yeah. The UEFI version will boot natively. So I would...

Leo: You'd have to get it signed.

Steve: I'll either get it signed, I mean, it's probably very much like the driver signing process we have now. As we know, Windows 10 requires signed drivers. And I have a driver that I created as a little side project for something that Lorrie needed that needed to run under Windows 10. So I got myself certified and got a driver signed so that it would run under unmodified Windows 10 systems. So I imagine it's sort of like that. I'm sure I will be able to do that.

Leo: It's like getting an extended certificate or something. It's really just to prove you are who you say you are.

Steve: Yeah. And I may be able to do the same sort of shim thing where I get them to sign my CA, which then securely boots, in Secure Boot, boots SpinRite onto the system.

Leo: Right, right. And there are a lot of Linux users that don't use GRUB to boot, by the way. There's other boot managers. Systemd is a very popular one.

Steve: Right.

Leo: I think GRUB is kind of probably fading away.

Steve: Well, boy, I mean, it got I guess a dearly needed security update. Wow. Whoo.

Leo: rEFIt is also popular. All right, my friend. That's it for today. Good job. Your little boot with a hole in it. Steve Gibson does this show every Tuesday - I usually show up - about 1:30 Pacific, 4:30 Eastern, 20:30 UTC.

Steve: Especially now that you're grounded, Leo.

Leo: I ain't going nowhere. That's right. I'm here, man. You can join us live if you want, watch us make the show. That's easy enough. All you have to do is go to TWiT.tv/live. There's live audio and video streams from a variety of sources there. Pick the one you like. No more Mixer, but we still have others. You can also get the show after the fact. Steve's got 16Kb audio. You're going to stop doing that, you said?

Steve: No, I think it's popular, yeah.

Leo: Why not? He also does the transcripts, which are great, and of course 64Kb audio. So those versions are all at GRC.com. While you're there, pick up a copy of

SpinRite. Hey, it couldn't hurt. Great system recovery tool, hard drive recovery and maintenance utility. Everybody ought to have it. And if you get it now, you'll be ready to get 6.1 the minute it comes out, plus all the interim releases Steve's working on. Somebody says your next release will be, what did they call it, something to bypass - BootRite, Steve's next program to fix the Secure Boot issues. I like it. BootRite. GRC.com.

We have the show also, audio and video, at TWiT.tv/sn. It's also on YouTube. You can subscribe in your favorite podcast application. That'd be the best way. That way you'll get it the minute it's available, each and every Tuesday afternoon. Thank you so much, Steve. Have a great week. Stay safe. We'll see you next time on Security Now!.

Steve: Will do. Bye.

Leo: Bye.