WannaCry Aftermath

Description: This week we examine a bunch of WannaCry follow-ups, including some new background, reports of abilities to decrypt drives, attacks on the kill switch, and more. We also look at what the large Stack Overflow site had to do to do HTTPS, the WiFi security of various properties owned by the U.S. President, more worrisome news coming from the U.K.'s Theresa May, the still sorry state of certificate revocation, are SSDs also subject to Rowhammer-like attacks, some miscellany, and closing the loop with our listeners.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-613.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-613-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We have a, I would say, a potpourri show, everything from a silly riddle to the original diagram of Ethernet from 1973. And a surprisingly young picture of Steve Gibson from 1984. That and a lot more, including an analysis of the WannaCry aftermath, all coming up next on Security Now!.

Leo Laporte: It's time for Security Now! with Steve Gibson, Episode 613, recorded Tuesday, May 23rd, 2017: WannaCry Aftermath.

It's time for Security Now!, the show where we cover your security and privacy online, rapidly becoming the most popular show on our network. And I have to credit this guy right here, our security guru, Steve Gibson of the GRC Corporation. We were at - hi, Steve.

Steve Gibson: Hey, Leo, great to be with you.

Leo: We were at the Maker Faire.

Steve: Oh, didn't that look like fun?

Leo: It was fun.

Steve: I was just drooling, looking at all of the stuff there.

Leo: Well, a lot of people came by to say hi. And to a person they said, "You want to know what my favorite one is?" And I said, "Yeah, I guess." Security Now!, they all love Security Now!. Maybe it's because it's the geekiest show we do. Makers like Security Now!, I guess. Anyway, take it for what it's worth, a data point.

Steve: Well, thank you. I appreciate the feedback.

Leo: Well, I love the show, that's for sure.

Steve: That's great.

Leo: And today we're going to WannaCry some more.

Steve: Oh, well. Naturally there's been lots of, you know, with anything as big as WannaCry, it doesn't just end like last Tuesday when we did the show and sort of pronounced it done. There's more. So as I was pulling everything together, I realized about half of what we had to talk about was still WannaCry. So the first half of the podcast we'll discuss the aftermath and various things. Like there have been claims that there's a way to decrypt it. There's questions about does it actually infect XP. And even some interesting reporting, I think it was from the Washington Post, we'll get there in a second, about the NSA's position on this, which - and I've got some interesting feelings about that, as well. So we'll examine a bunch of WannaCry follow-ups, including some new background, reports of the ability to decrypt drives, attacks on the kill switch and more.

We'll also look at what the large Stack Overflow site had to do in order to implement HTTPS. It's a beautiful sort of case in point of a big, sprawling, complex site and how it turns out not to be as simple as just using Let's Encrypt. Then there's an interesting question about the WiFi security of various properties owned by the current U.S. President. And you can imagine it's not good. In fact, it's very worrisome.

And speaking of worrisome, there's news coming from the U.K.'s Theresa May about the continuing plans - the headlines are odd. They talk about the U.K. wanting to create a new Internet. And it's like, what? And then - which is like a strange headline. But it turns out what they want to do is to basically really put the screws to the Internet that we have. And so what the people doing the headlining were trying to imply was it's so onerous that it's a different Internet. It wouldn't be like the one we have now.

Leo: Oh, I see.

Steve: Yeah. I also want to talk about the still sorry state of certificate revocation, which of course we dipped into deeply a few years ago when I realized what Chrome was not doing, and I created the revoked.grc.com domain specifically to demonstrate and allow users to test the revocation awareness of their browsers. The revoked.grc.com domain is a deliberately revoked certificate. And the question is, can your browser go there or not?

And Chrome just happily does, even today. But there have been additional movements that I want to sort of catch everybody up on. Then the question of whether SSDs are also subject to Rowhammer-like attacks. And some research suggests that there are two different types of attacks that can be employed against SSDs to damage them deliberately.

And then of course we've got some miscellany and some closing the loop with our listeners. So I think another jam-packed, information-laden podcast.

Leo: Of course. I expect nothing less for you, my friend. I've got your Picture of the Week ready to go, Steve.

Steve: Ah. So, yes. Yesterday Bob Metcalfe, the inventor of Ethernet, tweeted that it was the 44th anniversary of his invention of Ethernet. And so our Picture of the Week, he also included in his tweet, was a hand-drawn, because this was 1973, this was the year I graduated from high school, he was inventoring...

Leo: He was inventoring.

Steve: Inventoring, yes. He was at the Xerox PARC, the Palo Alto Research Center, PARC. And so this was his hand-drawn note. And I was going to say that back in '73 we didn't have, like, the kind of computers...

Leo: You didn't have Vizio or Trello or anything. You've got to...

Steve: Exactly.

Leo: This is a napkin, my friend.

Steve: Exactly. And so he had a typewritten description of it, but he needed a graphic to go with it. And I just love this picture because in the upper left he shows like a bunch of Altos, which was the GUI workstation which Steve Jobs famously saw and said, okay, that's the way you do these things. And so he has sort of this kind of stick figure-looking network with Altos hung onto it in different places. There's a Dynabook. There's a Nova, which was a popular minicomputer of the day. There's a PDP-11. And then he has it labeled with an arrow pointing to it, a "cable-tree ether," meaning a cable-tree conductive medium. And then in the second diagram to the lower right he shows, again, a cable Ether, and then what he calls a "telephone coax booster" with a pair of copper lines. And he points to it, saying that that's a "telephone ether." And then he's got a wacky little thing that this telephone ether connects to that says "telephone radio booster" and a little, looks like a cactus, but it's meant to be a radio antenna sticking up. And that's his radio ether.

So, I mean, he foresaw WiFi back in 1973. I mean, this is exactly what we have today. And I will come back to this in our Miscellany because I want to talk about the conceptual leap that he made, that he had to make, and why it was so different than the solutions of the time, and how very much like the original concept of the Internet, which we've

discussed before, how counterintuitive it was that you could autonomously route packets with no guarantee for their delivery. That is, the system itself couldn't guarantee delivery, but it just made a best effort, and how that was enough to revolutionize the world's communications. And so this is sort of a - this is a small, down at the connectivity part, but it's a similar innovation. So anyway, just a neat photo this week. I'm so glad that Bob gave me the opportunity to talk about it. And he and I have met on a number of occasions. He's a great guy and a real - an engineer's engineer.

So the first thing I wanted to note was, and it was the Washington Post had some interesting coverage and some background about the NSA's reaction to and feelings about the loss of their control of this very potent hacking tool, which was turned into, as we know, this WannaCry worm. The Washington Post writes: "When the National Security Agency began using a new hacking tool called EternalBlue, those entrusted with deploying it marveled at both its uncommon power and the widespread havoc it could wreak if it ever got loose. Some officials even discussed whether the flaw was so dangerous they should reveal it to Microsoft, the company whose software the government was exploiting, according to former NSA employees who spoke on the condition of anonymity given the sensitivity of the issue.

"But for more than five years, the NSA kept using it through a time period that has seen several serious security breaches and now the officials' worst fears have been realized. The malicious code at the heart of the WannaCry virus" - which we actually know is a worm - "that hit computer systems globally late last week was apparently stolen from the NSA, repackaged by cybercriminals, and unleashed on the world for a cyberattack that now ranks as among the most disruptive in history.

"The failure to keep EternalBlue out of the hands of criminals and other adversaries casts the NSA's decisions in a harsh new light, prompting critics to question anew whether the agency can be trusted to develop and protect such potent hacking tools. Current and former officials defended the agency's handling of EternalBlue, saying that the NSA must use such volatile tools to fulfill its mission of gathering foreign intelligence. In the case of EternalBlue, the intelligence haul was 'unreal,' said one former employee. 'It was like fishing with dynamite,' said a second. The NSA did not respond officially to several requests for comment for this article."

And then the story goes on to reiterate a bunch of background that we've already discussed. And as I said last week, I understand that this is an extremely tough call. My feeling is it's easy for those on the sidelines to jump up and down and say that the NSA should not secretly develop and then keep secret such powerful vulnerabilities. But I think it's very important to note that what is unfortunately missing from all of the reporting of this story are any details as a function, I mean, as a consequence of the nature of the fact that they have to be kept secret, of what the use of this longstanding SMB Windows vulnerability may have been to U.S. national security during the time that it was both available for the NSA's use and secret. We don't know how useful it was. We quoted, or the Washington Post quoted that comment saying that "the intelligence haul was unreal." But we have no details. We don't know what valuable intelligence it may have uniquely allowed to be gathered. And of course, had that been patched, that asset, that powerful intelligence-gathering asset would have been killed.

So horrific as its escape doubtless was, maybe if we knew how our intelligence services had been able to use its unique powers during the time of its availability, we might feel differently. But we're unlikely to know one way or the other. My only point here is to observe that we do not have the benefit of the full story. We don't know that it might change our judgment if we did know. So I thought - I appreciated the Washington Post reporting on that, and a little hint that it really was such a powerful tool.

And on the heels of this comes the proposed PATCH Act, a new bill designed to prevent occurrences like WannaCrypt. And I'll explain what it is and give some background and why I think it's a total crock. Okay. So SecurityWeek reported on this, and it was picked up from additional coverage, but basically a repetition of what is known, which is there is a bill now. So SecurityWeek writes: "Following the worldwide WannaCrypt ransomware attack that leveraged the EternalBlue exploit developed by and stolen from the NSA, Microsoft's chief legal officer" - as we discussed last week and quoted Bill saying - "called for governments to stop stockpiling zero-day exploits. His arguments are morally appealing, but politically difficult.

"Now, however," writes SecurityWeek, "he has partial support from a bipartisan group of lawmakers." And there's a bunch of them: Senators Brian Schatz, who's a Democrat from Hawaii; Ron Johnson, a Republican in Wisconsin; Cory Gardner, Republican from Colorado; and U.S. Representatives - those were senators - U.S. Representatives Ted Lieu, Democrat in California; Blake Farenthold, a Republican in Texas. Schatz announced yesterday that they had introduced the - and this is where PATCH is an acronym, actually - Protecting our Ability To Counter Hacking Act...

Leo: Oh, please.

Steve: I know.

Leo: They must spend more time on that than the actual bill. I swear to god.

Steve: I know. Well, and just wait because, I mean, it's so useless - of 2017. "Its purpose is to establish a Vulnerability Equities Review Board..."

Leo: That already exists. Obama set that up. Didn't work.

Steve: Exactly, "...with permanent members including the Secretary of Homeland Security, the Director of the FBI" - whoever he is - "the Director of National Intelligence, the Director of the CIA, the Director of the NSA, and the Secretary of Commerce, or in each case a designee thereof. Its effect will be to seek a compromise between the moral requirement for the government to disclose vulnerabilities," and then in parens SecurityWeek writes "(Microsoft's Digital Geneva Convention), and the government's political expediency in stockpiling vulnerabilities for national security and deterrence purposes.

"In a statement issued yesterday, Schatz wrote, 'Striking the balance between U.S. national security and general cybersecurity is crucial, but it's not easy. This bill strikes that balance. Codifying a framework for the relevant agencies to review and disclose vulnerabilities will improve cybersecurity and transparency to the benefit of the public, while also ensuring that the federal government has the tools it needs to protect national security.'" Okay, except that, if what they're suggesting is that EternalBlue should have been made public five years ago, obviously there would be huge NSA pushback against that.

So anyway, continuing: "The bill does not go so far as to mandate the disclosure of all government zero-day exploits to relevant vendors for patching, but instead requires the

Vulnerability Equities Review Board to develop a consistent and transparent process for decision-making. It will create new oversight mechanisms to improve transparency and accountability, while enhancing public trust in the process. It further requires that 'The head of each federal agency shall, upon obtaining information about a vulnerability that is not publicly known, subject such information to the process established.' In this way the Vulnerability Equities Review Board" - it's hard to even say that.

Leo: That exists, by the way. So this must be some sort of update to this.

Steve: Right, "...has oversight of all zero-day vulnerabilities held by the government agencies. It also maintains the controls relating to whether, when, how, to whom, and to what degree information about a vulnerability that is not publicly known should be shared or released by the federal government to a non-federal entity, that is, whether the public interest requires the vendor be able to patch the vulnerability."

Leo: It's a balancing act, obviously, because if you're going to have spy agencies, they're going to collect these.

Steve: Well, and my take is this is total nonsense, a bunch of politicians who want to appear to be responding by creating more government bureaucracy - which is exactly, by the way, what our present administration was elected to reduce.

Leo: Right.

Steve: This will simply add regulation without effect. The CIA and NSA will loudly and probably honestly assert their need for these for national security. They will downplay the downside and explain how we're already losing the cyberwar, and how forcing voluntary disarmament would be unilaterally laying down our arms and capitulating in the cyberwar. They'll argue that foreign governments who lack the PATCH Act's attempted oversight controls will still be free to discover and use the very same software flaws against us; and that, as we have seen, even when patches were already made available, machines were still victimized. That is, you know, Microsoft patched this in March, and this happened, what, two weeks ago.

And they'll also note correctly that Microsoft only back-patched XP and Vista because of and as a result of the proven severity of the problem. Which would argue that, if they had informed Microsoft, oh, by the way, here's an SMB flaw, Microsoft certainly would have fixed the OSes that are widely in use and recent, but would never have bothered to fix XP and Vista. They did that only because it was such a problem. So that argues that even informing Microsoft of bad problems doesn't guarantee that all of the machines online will get fixed. So I would argue that this is just bureaucracy with no effect.

Leo: I think there has to be some discussion. I mean, unless you want to abolish antiterrorist efforts, I don't think it's unreasonable for the government, for the NSA and the CIA, to stockpile exploits; right? I mean, we're not saying that. And then it's also reasonable, and I think the NSA knows this full well, it was the Director of National Intelligence who set up the original vulnerability equities process in 2008.

They understand the risk also of holding on to these. And so I think it's not inappropriate for some group of people who represent both the public interest and the interests of the intelligence community to somehow hash this out. I mean, otherwise you just keep everything.

Steve: Well, if it comes down to money, and as we know, so much often does, then if nothing else maybe this - okay. And if putting more money into protecting the secrets would have prevented that leak, then maybe that's the solution, that is, the fact that the NSA lost something that generated headlines, that has been such a huge problem. If they needed more money in order to - a budget in order to further and better secure their secrets, that's what they should ask for and get.

Leo: Yeah, that's a good solution, yeah.

Steve: Right.

Leo: But, I mean, it's just as hard to make perfect software as it is to make a perfect spy. And I think this is real world. There's no obvious right answer. But I think it's appropriate to try to find an answer. I'm not sure how this differs from the existing vulnerability equities process. But this is a...

Steve: Well, and a bunch of guys sitting around saying, oh, we found a really juicy one, we need to keep it. How is anyone going to say, no, you have to tell?

Leo: There's some evidence that Heartbleed, the NSA knew about Heartbleed for several years before it was discovered. It never leaked out. It was discovered. Do you want, I mean, these are tools the NSA uses, not for - one of the things is this is generally, as far as I know, not used, ever used for mass surveillance. It's used for targeted surveillance.

Steve: Because they don't want it to get out.

Leo: Right.

Steve: Yeah. So they're going to find someone somewhere and use it in order to perform a network penetration. And, yeah, I just - I think it's going the way it should. But I guess my point is the idea of this being a proper subject of a committee is ridiculous. To me, this doesn't make sense for a bunch of bureaucrat heads of departments to sit around and say, gee, I mean, the worse it is, the more powerful it is, and so the more useful it is. The mechanics just don't make sense. So, I mean, certainly if the NSA were to find something that they thought they should disclose, they would. They would say, hey, Brad - I meant to say Brad. I said Bill earlier. Brad Smith, here's a goodie for you. You probably just need to patch this soon. And they would. But that's - it seems to me it's the discoverer needs to react responsibly, and no committee should reasonably expect to be able to impose their own judgment on...

Leo: Well, I'm not sure that's true because there are, I mean, this is the existing VEP. And they have to answer questions like how much is the vulnerable system used in the core Internet infrastructure, in other infrastructure systems in the U.S. economy? Does the vulnerability, if left unpatched, impose significant risk? How much harm could an adversary or criminal group do with knowledge? These are appropriate questions. And I think who else is going to answer these questions? You've got to sit down with people who both want to protect, you know, understand the issues and discuss it. How badly do we need the intelligence? Are there other ways we could get it? These are all questions that were supposedly going to be asked in the original process.

Steve: Those questions are great and completely useless.

Leo: You're assuming bad will on the part of the spies, who are going to say, no, no, you can't have anything. But I think...

Steve: No, I'm not. I'm assuming self-interest. I'm assuming that they're representing, I mean, they understand how difficult it is to find a problem like this and how valuable it is to their needs. And I just don't think them explaining it to a panel of other agency heads is going to have any useful result. They're going to say, "Yeah, this is really bad, which makes it really important, really valuable." So it's like, I don't know, just the concept of discussing it just to me seems...

Leo: Well, there you're giving up because they're just going to keep it. If there's no process at all, then there's no challenge to them just keeping it. Then you're just giving up and saying, well, they're going to have it. Nothing you can do about it.

Steve: Yeah. And, see, I think the challenge is fake. I think it's a fake challenge. I think it's a made-up bureaucratic circle jerk.

Leo: Just to show that they care, even if they don't.

Steve: Yeah. Yeah. So there's a great cartoon on the next page of the show notes, Leo, that is the lead-in to this WannaCry kill switch. We discussed that it existed, and it didn't make sense to us. We now believe that we understand what it was for. And it was of dubious value. It turns out that when malware is being forensically reverse-engineered, it's put in a forensics sandbox. And malware often makes DNS queries out to its command-and-control servers. And so this forensic sandbox responds to the DNS queries with its own local server IP in order for the malware to attempt to connect to a command-and-control server.

So a way for malware to detect whether it is sandboxed and being forensically examined is to just ask for a gibberish DNS domain that doesn't exist. And if it cannot get a connection to a server at that DNS domain, it'll just assume it's not in a sandbox and do its dirty work. But if it does get a connection to a domain it knows it just made up, it was all gibberish characters - remember I insisted on reading most of that domain name last week - then it goes, oh, no, I must be in a sandbox, and so it alters its behavior.

Leo: Smarter way to do this would not be to have a hardcoded domain but just a random string each time; right?

Steve: Exactly. Exactly. There's no way...

Leo: Which makes me wonder if this is really the rationale behind it. But maybe they really wanted a kill switch, like a real...

Steve: Yeah. This cartoon is wonderful because it shows some bad guys in masks with a skull-and-crossbones behind them, saying, "Look, I included a few lines to check whether we've been sandboxed, so white hats can't detect us." And so the other guy says, "I'll just ping a nonexistent domain." And then, "If the nonexistent domain responds, it means we're in a sandbox, so we won't encrypt any files. That way no one will notice us." And then the first guy says, "Ha ha, well done. We're going to get so many bitcoins." Then, "a few hours later," our now famous - unfortunately famous, he's not happy, by the way, how famous he is. We'll get to that in a second. He says: "Huh, what's this inactive domain hardcoded in their code? I should reserve it. You never know." And then in the final frame it says "Sh*t" when these guys realize, ooh, we've been foiled. And of course...

Leo: Curses. Foiled again.

Steve: This thing completely neutered as a consequence of registering - you're right, Leo. If they'd just used a random number, just used the little CryptoAPI that they were already using to give them a chunk of entropy, convert that to seven-bit ASCII, stick a dotcom on the end of it and send it out, then yes, then something - then this very simple trigger wouldn't have happened. But that's the best theory for what they were trying to do.

Leo: Unless they really wanted a kill switch. There might have been other reasons they wanted a kill switch.

Steve: Yeah, yeah.

Leo: Who knows?

Steve: So, meanwhile, what that creates is a server answering TCP connections at a known domain, and thus a known IP. So what's the next logical thing to happen? Hackers now trying to reignite WannaCry with nonstop botnet attacks. In other words, the kill switch server is now being DDoSed continually by various Mirai and Mirai-derivative botnets in order to force it offline so that, when the WannaCry worm reaches out to see if it should encrypt or not, if it is unable to get a connection because the server, the kill switch server is being DDoSed, it'll go, oh, no one there, so off we go to encrypt the files on the drive.

There are a few problems with this. First of all, only newly infected or rebooted systems propagate. So a new infection or a rebooted system, when the worm comes back alive from a reboot, they spend only the first 24 hours scanning for other vulnerable machines. Then they stop. So the point is it's not as if putting the kill switch server under a persistent DDoS would immediately bring the WannaCry, the existing infection base of WannaCry back to life. Instead it would only - they'd have to keep it offline, and then only new infections or rebooted systems would then reach out to see whether the kill switch server is available and attack. So it's like, yeah, okay, fine.

And, by the way, the company is working, the company behind the guy who found this, an L.A.-based security firm, is now gone to an unnamed DDoS protection service. And I didn't bother to check to see. It's easy, I mean, it's unnamed, but it's easy to figure out who it is because their servers are going to answer the IP for that domain. And so they're doing everything they can to help keep the TCP server at that domain responsive in order to keep the original variant of WannaCry from propagating. As we know, immediately that code with hex edited out and another version launched. And there have been, I think now the last count I heard was five variants of this. So, and then also I didn't put it in the notes, but there's even another much different variant which uses six different exploits from the NSA, rather than just two. Rather than just EternalBlue and DoublePulsar, this uses a whole bunch of them. So the feeding frenzy has begun.

And so the question is, who is MalwareTech? Who is this person? Unfortunately, he's in the U.K., and the U.K. has rather notorious reporting. He apparently tried in vain to remain cyber and not identified in the physical world. Nowhere on either his Twitter page nor in his blog are there any names, details, headshots, anything that would connect him to the real world. Which should make it clear that he wished to remain anonymous. But he was dogged by British tabloid reporters who dug deep into his entire online past and finally outed or "doxed" him as being a 22-year-old British security researcher named Marcus Hutchins.

On the other hand, thanks to being known, the ethical hacker group, HackerOne, has awarded him $10,000 for his efforts, and Marcus stated that he intends to split that award between charity and educational resources for students who cannot afford them. In the show notes here I have a snapshot of several of his posts. He tweets as @MalwareTechBlog. And so he tweeted: "I knew five minutes of fame would be horrible. But honestly, I misjudged just how horrible. British tabloids are super invasive." He tweeted: "Journalist doxed a friend, then rang them, offering money for my girlfriend's name and phone number. One turned up at another friend's house." He tweeted: "Tabloids here don't care about the story, they care about every detail of the person behind it and will go to extreme lengths to find out." And then, finally, he said: "One of the largest U.K. newspapers published a picture of my house, full address, and directions to get there. Now I have to move." So that's the world we live in today, unfortunately.

Okay. On the question of decrypting WannaCry, there's been a lot of confusion and misinformation, a real sort of a development frenzy to see whether there was a way to decrypt. And now there's some sanity that has come to this, and things have settled down, and we sort of know what's going on. There's a function in the Windows cryptographic API called CryptReleaseContext. So the idea is that, when you're going to start doing a bunch of cryptographic work, you acquire a cryptographic context from Windows. And so Windows gives you a handle, and then you apply different actions, different methods, different functions against that handle.

So the handle is an abstraction of a cryptographic context, the idea being that Windows is supposed to handle all of the messy business for you. So you say things like "Create asymmetric key," and it just does. And then you say, "Use that key to do this or do that

and so forth." And so Windows provides this interface and deals with it all. Well, when you're done, you want to explicitly destroy any sensitive information that may have transiently existed. So you release the context. You say to Windows, here's that handle, that context handle you gave me. I'm releasing it back to the system. You hope that Windows proactively wipes and zeroes all of the memory that was involved in any sensitive operations.

It turns out, believe it or not, that - I'm not sure about Windows 8. But at least we know Windows 10 does, but Windows 7 does not. So XP, Vista, and 7, the API up through 7 at least does not proactively wipe the memory. As a consequence, sensitive memory contents containing both the symmetric key and even the two primes which were originally obtained in order to compose the private key, the RSA private key, have been found after the fact in memory from XP through Windows 7. So if an XP, Vista, or Win7 machine is encrypted with WannaCry, and if it's not powered down, it's very likely that the key can be found.

And there is now - it looks like the best software, I've got links to the various GitHub projects, there's one called WanaKiwi, W-A-N-A-K-I-W-I. That looks like it's the best and easiest to use. You can simply run it, wanakiwi.exe. You just run it. It will find the process, scan the process memory, find the private key, find the primes, reconstruct the RSA key from them, and then use that in order to return all of your files, to decrypt all of your files. Obviously it's with lots of limitations. You can't have used the machine a lot, or that increases the probability of something overwriting that memory, which was released back to the operating system. It should have wiped it. Windows 10 does.

But apparently, we don't know about 8.1, but Windows 7 is known not to. And in their testing of it, under XP, every attempt to decrypt when the machine had not been powered down - and I'm not sure about rebooting, whether that would wipe the memory. It doesn't necessarily wipe the memory when you reboot. But that probably makes it much more difficult to find. Within these limitations, it looks like it's possible. But that's as a consequence of these developers using the CryptoAPI.

Leo: So funny, it's another Windows bug that's being used to mitigate.

Steve: Yup.

Leo: Father Robert's going to do it on Know How. He's already infected a machine. We have a little mini network. Last week he WannaCried it. This week he's going to WannaCry it and see if the disinfection works.

Steve: Cool. Nice. And, for example, this is one of the reasons I'm not using any unknown functional CryptoAPI in the SQRL code.

Leo: NACL, baby.

Steve: It's all my own, yeah.

Leo: You're not even using NACL?

Steve: I used Bernstein's crypto stuff. But I maintain my own security region, which is swap locked, so it can't be swapped out. And my code is constantly zeroing that region. The moment it's through with anything sensitive, it proactively wipes it. And I've got various monitor daemons that are always checking to make sure that nothing happened that allowed anything to stay there. So I've been, I mean, that's the way you write secure code is you, from the first moment, you are scrupulous about the way you manage the secrets in your code.

Leo: Not Microsoft, apparently.

Steve: No. And our friend Simon Zerafa tweeted: "98% of WannaCry victims running Windows 7. Can we finally nail the lid shut on the XP myth?" And in the show notes here is a picture that Kaspersky created and that BleepingComputer quoted, showing the various infection levels. What we know is that XP can be infected, but the worm doesn't propagate. The worm's SMB attack fails on XP, but it does succeed on Windows 7. So the reason so many Windows 7 machines are infected is actually a mistake in the packet composition for the exploit that didn't infect XP. So, yes, we have closure on that, too.

Leo: All right, Steve. We continue on.

Steve: So how many infections? The latest count from the WannaCry sinkhole, the kill switch server, is 416,989 individual IPs. Now, apparently that's not including the 604,102 unique IPs from manual visits to the domain, so non-worm visits, just people curious.

Leo: I've done that. I typed it in.

Steve: I'll just go check that crazy...

Leo: See what's going on there.

Steve: Yeah. So just shy of 417,000 infected machines. Wow, that's a biggie.

Leo: Yeah.

Steve: The Wall Street Journal, of course, has their content behind a pay wall. And there was a blurb in Apple News that I saw yesterday, but I didn't read it there at the moment. And now I can't get to it, and I'm unwilling to give - I just don't use The Wall Street Journal enough for it to make sense for me to pay for what's there. But I just thought - the title of the article was "All IT Jobs are Cybersecurity Jobs Now." And the subhead was "The rise of cyberthreats means that the people once assigned to setting up computers and email servers now treat security as top priority." And I just, even though that's all of

the story that I have, that's really as much as we need for this podcast.

Leo: Oh, neat, yeah.

Steve: Yeah. And I just thought that was an interesting sign of the times, and of course I think really good news, that rather than there being, like, some guy whose job is security - we've talked about this, that it's not something that one person can be responsible for in a huge organization. All of IT, everybody has to have at least that in the back of their mind, at least an awareness of the security consequences and implications of everything they do. And, sure, you could still have a czar who keeps reminding them and refreshing them. Someone sent me a note over the course of the last week that their company actually sets up fake external pseudo entities that send mail into their company, trying to trap their own employees into clicking on a link. And then, if it works, they get a visit from someone in the corporation saying, you know, yesterday you got this email, and you clicked on it, and we need to have a talk because you can bring down the entire organization.

Leo: Yeah, yeah, isn't that great.

Steve: So, yay. Okay. So Stack Overflow. As a developer, when I'm in a hurry, and I just want a quick answer to something, I google a phrase, and more often than not Stack Overflow is the first five responses. And the content is not super high quality. There's a lot of people giving their opinions. But it's a starting point. And very often there will be a link to something that will be definitive. And it's like, oh, great. So it's, for me, I mean, I'm intimately familiar with Stack Overflow. They have a huge, you know, Stack Exchange is the family. And it's a huge and sprawling website.

So yesterday, Monday, yesterday, as I wrote here in the show notes, the well-known, very popular, and sprawling Stack Exchange family of developer-oriented websites completed their conversion to HTTPS. Nick Craver, who's a software developer and systems admin for Stack Exchange, who spearheaded this work, yesterday detailed the journey in a very lengthy blog post. I have the link here in the show notes. And I read through a lot of it. Nick is sick and tired of everyone saying "Just use Let's Encrypt," as if that would simply solve every problem. So his detailed posting explains and will be of interest to anyone who wants to gain a better understanding of why "Just use Let's Encrypt" is not the answer for every case.

To get some sense for the nature of the challenge, Nick had a bulleted summary, noting that, "For example, we have hundreds of domains, many sites and other services, many second-level domains - stackoverflow.com, stackexchange.com, askubuntu.com, and so on. Many fourth-level domains, for example, meta.gaming.stackexchange.com." He says: "We allow user submitted and embedded content, images, and YouTube videos and posts. We serve from a single datacenter. We have ads and ad networks. We use websockets, north of 500,000 active back to our server at any given time. We get DDoSed, so we have to have proxies. We have many sites and apps communicating via HTTP APIs." And he says in parens, "(creating proxy issues). And we're obsessed with performance." And he says in parens: "(Maybe a little too much.)"

So anyway, to summarize, he says: "The most common question we get, why not use Let's Encrypt?" And he says: "Answer: Because they don't work for us." That is, Let's Encrypt won't work for us. "Let's Encrypt," he says, "is doing a great thing. I hope they

keep at it. If you're on a single domain or only a few domains, they're a pretty good option for a wide variety of scenarios. We are simply not in that position. Stack Exchange," he writes, "has hundreds of domains. Let's Encrypt doesn't offer wildcards. These two things are at odds with each other. We'd have to get a certificate or two every time we deployed a new Q&A site, or any other service.

"That greatly complicates deployment and either, A, drops non-SNI clients" - you remember that's Server Name Identification that allows multiple domains to reside at a single IP. And he notes that, even today, around 2% of their traffic are non-SNI-aware clients. So they're one in 50. "Or, B, requires far more IP space than we have." Meaning that they would need, in order to accommodate that, they'd need to bind individual certificates to IPs, and they don't have that many IPs.

And he says, "Another reason we want to control the certificate is we need to install the exact same certificates on both our local load balancers [which terminate the - and I'm adding this - which terminate the TLS connections and then forward the traffic to the servers behind them] and our CDN proxy provider. Unless we can do that, we can't fail over, that is, away from a proxy, cleanly in all cases. Anyone that has the certificate pinned via HPKP" - which is HTTP Public Key Pinning - "would then fail validation. We're evaluating whether we'll deploy HPKP, but we've prepped as if we will later."

And then he finally says: "I've gotten a lot of raised eyebrows at our main certificate having all our primary domains plus wildcards. Here's what that looks like." And I have a picture of it in the show notes, and I had to smile when I saw who Stack Exchange chose to build for them an amazing certificate. And our listeners can probably guess: It's DigiCert. And they produced an amazing certificate because it has a high-assurance extended validation. Oh, I'm sorry, no. The EV is an EV root, but it is also an SHA-2 high-assurance - I'm trying to decide if the Stack Exchange certificate is EV. Can you go to www.stackexchange.com, Leo?

Leo: Sure.

Steve: And see if you get an EV flag? Because I'm assuming, since it's signed by - they have a high-assurance intermediate and the EV root.

Leo: Yes, it's green. That means it's [crosstalk].

Steve: And so you can't normally get that. I mean, EV certs, I mean, they had to go to some serious hoop-jumping in order to create a certificate like this. You can see...

Leo: Yeah, can't do wildcard, there's all sorts of, I mean, this is not - Let's Encrypt never promised to be the ultimate solution that would put everybody else out of business.

Steve: Correct. And then in Nick's Q&A he asks himself the question: "Where do you get certificates?" And then he answered: "We use DigiCert. They've been awesome."

Leo: Yeah, so do we. That's what TWiT.tv is.

Steve: And everyone knows I do, too, yes. So they got DigiCert to make them a very special extended validation certificate with a massive SAN, which we were also talking about recently because remember that there's going to be a deprecation of the use of the certificate's name in favor of only domains explicitly listed in the Subject Alternative Name, the SAN record, which in this case enumerates all of the various domains and wildcards for subdomains underneath those domains. And of course issuing that would have taken a great deal of careful work on DigiCert's part because they needed to firmly, and with EV veracity, verify and prove ownership of every root in that second-level domain set.

So bravo to Stack Exchange for making the effort, a significant effort to go to HTTPS, and DigiCert for providing them with the ability to do that, essentially. And, yes, as you say, Leo, and as he explained, Let's Encrypt is not the simple answer except for simple certificates. So it achieves what they were trying to achieve, which is lowering the bar and making it difficult for someone to say, oh, we are not HTTPS because we can't - doesn't make sense for us to afford a certificate. Well, now simple DV, simple server certs can be free.

Okay. So ProPublica and Gizmodo co-published this report on hacking Mar-a-Lago. I've got the links in the show notes, but I'll just share the top of it, which is kind of fun and summarizes it. They wrote: "Two weeks ago, on a sparkling spring morning, we went trawling" - in more ways than one - "along Florida's coastal waterway. But not for fish. We parked a 17-foot motor boat in a lagoon about 800 feet from the back lawn of The Mar-a-Lago Club in Palm Beach and pointed a two-foot wireless antenna that resembled a potato gun toward the club. Within a minute, we spotted three weakly encrypted WiFi networks." And I assume that means...

Leo: WEP; right?

Steve: ...WEP encryption. "We could have hacked them in less than five minutes, but we refrained." Of course they're publishing this publicly, so you bet they are saying they refrained. "A few days later, we drove through the grounds of the Trump National Golf Club in Bedminster, New Jersey with the same antenna and aimed it at the clubhouse. We identified two open WiFi networks that anyone could join without a password. We resisted the temptation.

"We have also visited two of President Donald Trump's other family-run retreats, the Trump International Hotel in Washington, D.C., and a golf club in Sterling, Virginia. Our inspections found weak and open WiFi networks, wireless printers without passwords" - we know how juicy those are - "servers with outdated and vulnerable software, and unencrypted login pages to back-end databases containing sensitive information.

"The risks," they write, "posed by the lax security, experts say, go well beyond simple digital snooping. Sophisticated hackers could take advantage of vulnerabilities in the WiFi networks to take over devices like computers or smartphones and use them to record conversations involving anyone on the premises. 'These networks all have to be crawling with foreign intruders, not just ProPublica,' said Dave Aitel, chief executive officer of Immunity, Inc., a digital security company, when we told him what we found." And of course that's the first thing I thought and our listeners would think is, yes, ProPublica is

not going to do anything that would create a felony cyber intrusion offense for them.

Leo: I'm surprised the Secret Service didn't come on over and say, "Hey, what you doing with that potato gun?" I mean, seriously.

Steve: Yes, I know. I know. I was watching something, I don't remember now who it was. It was some show where somebody resides next to Mar-a-Lago and, with a reporter and a camera crew, walked out onto the beach and walked over to Mar-a-Lago and walked right up onto the grounds.

Leo: I'm sure the President wasn't there at the time, though; right?

Steve: Yeah.

Leo: Don't you think there's a little better security when he's there? Not that I...

Steve: We would like to hope that. The problem is we already know that printers are easily taken over and used as a beachhead. And apparently there are open printers with no passwords. It's like, oh, lord. So, yeah.

"Security lapses," they write, "are not uncommon in the hospitality industry which, like most industries and government agencies, is under increasing attack from hackers. But they are more worrisome in places where the President of the United States, heads of state, and public officials regularly visit. U.S. leaders," they write, "can ill afford such vulnerabilities. As both the U.S. and French presidential campaigns showed, hackers increasingly exploit weaknesses in Internet security systems in an effort to influence elections and policies.

"Since the election, Trump has hosted Chinese President Xi Jinping, Japanese Prime Minister Shinzo Abe and British politician Nigel Farage at his properties. The cybersecurity issues we discovered could have allowed those diplomatic discussions and other sensitive conversations at the properties to be monitored by hackers." And of course the last thing we want in this case is for anyone to be blackmailed as a consequence of information that would be disclosed that way. Yikes.

And speaking of yikes, Theresa May is said to be creating a new Internet, to be controlled and regulated by the government. What we understand now, though, reading into this a bit, is that she is proposing to simply hyper-regulate what Internet connectivity they have control over. This was reported by the Independent.co.uk, saying that: "Theresa May is planning to introduce huge regulations on the way the Internet works, allowing the government to decide what is said online." And Leo, prepare yourself because I know how you're going to feel about this.

"The plans will allow Britain to become 'the global leader in the regulation of the use of personal data and the Internet,' the manifesto claims. The manifesto suggests the government might stop search engines like Google from directing people to pornographic websites. 'We will put a responsibility on industry not to direct users - even unintentionally - to hate speech, pornography, or other sources of harm,' the Conservatives write.

"Particular focus has been drawn to the end of the manifesto, which makes clear that the Tories want to introduce huge changes to the way the Internet works. 'Some people say that it is not for government to regulate when it comes to technology and the Internet,' it states. 'We disagree. In harnessing the digital revolution, we must take steps to protect the vulnerable and give people confidence to use the Internet without fear of abuse, criminality, or exposure to horrific content,' the manifesto claims in a section called 'the safest place to be online.'"

They write: "'Our starting point is that online rules should reflect those that govern our lives offline,' explaining the justification for a new level of regulation. 'It should be as unacceptable to bully online as it is in the playground, as difficult to groom a young child on the Internet as it is in a community, as hard for children to access violent and degrading pornography online as it is in the street, and as difficult to commit a crime digitally as it is physically.'" Yeah, good luck with all of that. That's me saying that.

"The manifesto also proposes that Internet companies will have to pay a levy like the one currently paid by gambling firms. Just like with gambling, that money will be used to pay for advertising schemes to tell people about the dangers of the Internet, in particular being used to 'support awareness and preventative activity to counter Internet harms,' according to the manifesto. The Conservatives will also seek to regulate the kind of news" - wow, fasten your seatbelt, Jeff Jarvis - "that is posted online and how companies are paid for it. If elected, Theresa May will 'take steps to protect the reliability and objectivity of information that is essential to our democracy' and crack down on Facebook and Google to assure that news companies get enough advertising money."

Leo: Ha.

Steve: What?

Leo: What?

Steve: "If Internet companies refuse to comply with the rulings - a suggestion that some have already made about the powers in the Investigatory Powers Act - then there will be a strict and strong set of ways to punish them."

Leo: Jeez.

Steve: "'We will introduce a sanctions regime [ugh] to ensure compliance, giving regulators the ability to fine or prosecute those companies that fail'" - I know - "'that fail in their legal duties, and to order the removal of content where it clearly breaches U.K. law,' the manifesto states. In laying out its plan for increased regulation, the Tories anticipate and reject potential criticism that such rules could put people at risk. 'While we cannot create this framework alone, it is for government, not private companies, to protect the security of people and ensure the fairness of the rules by which people and businesses abide,' the document reads. 'Nor do we agree that the risks of such an approach outweigh the potential benefits.'"

Leo: Wow.

Steve: I know. It's breathtaking. I mean, it is goodbye the Internet that we have known, and another example of personal responsibility being apparently just something that we shouldn't be responsible for.

Leo: Right.

Steve: Yeah, wow.

Leo: What you gonna do?

Steve: So an update on the state of certificate revocation. To sort of remind everyone where we've come from, the problem is that certificates are issued for a fixed time period. They have an expiration date typically no longer than three years, in some cases two, in some cases one, and in some cases even shorter, in the case, for example, of Let's Encrypt, where you have an automated issuance system, so it makes sense to shorten the expiration time because you've reduced the burden to near nothing to reissue. And that's one way of mitigating the problem with revocation.

The revocation problem is that, if a certificate is newly issued for, say, for three years, and then something happens that it gets out of control, the certificate's private key, which is essentially the certificate, signed by an authority, escapes the control of its owner. Then that would allow someone to maliciously use that in concert with a domain intercept or a traffic rerouting to take somebody to a site that is able to legitimately appear as the domain that that certificate covers. Or, horrifyingly, in the case of Stack Exchange, all the domains, the hundreds of domains that certificate covers.

So it was always recognized that there needed to be a means of managing revocation, that is, some way, once a certificate - because, like, the certificate is a static, self-contained assertion of its own validity. The statement it's making is that anyone who presents the certificate is the valid owner of the domain that is bound, the domain name bound in that certificate. And it has an expiration date. It has a not valid before/not valid after pair of date and time stamps. So it can exist independently and is assumed to be valid until the expiration date, unless something intercedes, thus revocation. It can be revoked by the CA, the Certificate Authority, but how is the knowledge of that revocation conveyed to somebody who wants to trust the assertion that the certificate makes?

Originally, we had what's known as a CRL, a Certificate Revocation List. Every certificate authority would publish a list and would maintain the list periodically. The problem is, as the Internet exploded, and as more sites wanted HTTPS and initially SSL and then TLS secure connections, we had an explosion of certificates. With that came an explosion in the size of the revocation lists. And the model of revocation lists was always kind of broken because it would mean that a user's browser that connected to a domain would get a certificate from the domain's server, would then look in the certificate for the URL of the signing authority's CRL server, and would ask that certificate revocation list server for the list of all revoked certificates. And this list could be huge.

Technically, once the certificate has naturally expired after its three-year life, plus some

slop for clock error, then that certificate could be removed from the list because the certificate has expired itself by virtue of its own - the time and date stamp that it carries. But still, that means that all certificates that have not yet expired, plus some fudge factor, would have to be on the list. And the model's broken because the browser just wants to check one domain. Instead, the model is ask a CA for every revoked certificate that isn't yet expired in a single list. And then the browser goes and searches it to see if the one it cares about is in the list. So that was all bad.

Chrome came along from Google, and of course they basically did nothing. Google went their own way and created what I actually revealed to be a wholly nonfunctional solution known as CRLSets. Which, when I looked at them, I realized, well, this can't work because it's basically a small list of highly publicized, high-profile revoked certificates. There were, like, seven of them when I first looked, plus a bunch of EV certificates that had been revoked. But back then most certificates weren't, and even today most certificates are not EV. Most of them are just DV, domain validation, not extended validation.

And so I created a revoked.grc.com cert, and it generated a lot of news. So Google added it. It was like Certificate No. 7 in their revoked list. And so I thought, okay, fine. And I created another one. And they haven't bothered to revoke that one because then I would just create a third. The point is Chrome doesn't do this, either. Completely broken. On the other hand, if you go to revoked.grc.com in Firefox, you get an error page that says "Security error: Revoked certificate." So, gee, Firefox is doing it. Chrome isn't.

Okay. So what was created then was an online facility. Instead of getting the whole list, the idea was, in what's called the OCSP model, the Online Certificate Status Protocol, the browser would fetch a certificate from a site it wants to have a secure connection to. Receives the cert. In the cert is an OCSP URL which gives it the domain of the server that serves the OCSP protocol. So now the browser simply asks for a specific domain. Is this domain valid? And the OCSP server says yes or no, depending. So that's, like, way better, but it's got some problems. If the OCSP server doesn't respond, then does it fail not trusting or fail trusting? Because if it fails not trusting, an outage of the OCSP protocol server would prevent you from accessing that site.

On the other hand, if it fails trusting, then bad guys could DDoS the server to get their certificate trusted or could block the OCSP traffic in order to get it to be trusted. So you're sort of messed up either way. Also, it creates a substantial additional roundtrip time. You've got to establish a TCP connection, bring up TLS because you need this to be a secure exchange, and then the communication. So it creates an additional time overhead and slows everything down. This is for every HTTPS connection that the browser would make to additional domains for the page. So it's better, but it's still got lots of problems.

The next idea was, oh, okay, let's have the server which is serving the certificate "staple," as the term is, a recently received OCSP assertion to the TLS transaction. So the idea being that the certificate lives a long time, but the OCSP assertion doesn't. And so periodically the server, the web server that wants to serve high-integrity certificates, it will go and get the OCSP assertion and include that in the handshake. That eliminates the overhead of the browser having to do it. It allows a long-lived certificate and short-lived reassertions that there's been no expiration so far, to keep it valid.

And so that's sort of nice except there's still a problem because the OCSP can't be in the signed certificate because that's digitally signed, and you can't change the certificate. So the OCSP staple has to be in the protocol, separate from the certificate. But that means if bad guys steal the certificate, they just won't provide a stapled OCSP, and so we're back

to the beginning again.

So the final solution was to embed in the certificate itself, the original signed certificate, a new flag, "OCSP Must Staple." So now the certificate says the server serving this must provide a valid OCSP staple, or the certificate is invalid. So that solves the problem of bad guys getting a certificate and just not including a staple, and also getting in the way of the OCSP and the need for the browser to go out and fetch its own OCSP information, thus slowing down all the transactions.

So all of that seems great, except where we are today is that neither of the two majority web servers on the Internet, which would be Apache with about a 46% share and Nginx with about a 20% share, neither of them have it correctly implemented. They are both badly broken. The idea should be that the server caches the most recent OCSP staple information and includes it with all the certificates that it hands out, like in the TLS handshake. And then periodically, as it begins to approach expiration, it reaches out to the OCSP server to refresh its cached OCSP assertion.

It turns out that both Apache and Nginx are horribly broken. If anything happens that causes an invalid OCSP to be received, or no OCSP to be received, rather than saying, oh, shoot, it's a good thing I started asking early, I'll wait a bit and try again, and again and again and again and again, hopefully finally achieving success before the existing staple expires, instead Nginx and Apache, if anything happens to their attempt to obtain a refresh, they invalidate the still-valid OCSP staple in their cache, thereby failing all connections to any server whose certificate has OCSP Must Staple in the certificate in order to succeed. So, okay, that means nobody can even today actually use OCSP Must Staple until the majority web servers on the Internet get themselves fixed. And people have tried to turn it on and immediately started having problems.

So right now - oh, and by the way, this problem was reported in 2014 to the Apache group, and it remains unfixed today. So apparently they've got other things to do, or it just isn't high enough on their priority list, or they don't care. I don't know. But it's always been a problem, I mean, the problem of revocation has always been known, and we've never yet come up with a good fix. We're getting closer. But one of the enduring lessons of this podcast is these things which are broken or are trying to evolve just happen on a very slow time scale.

Firefox is considering giving up on OCSP for DV certs. So this is the last shoe to drop on the OCSP side. They filed a bug report, just because that's how they sort of communicate, saying that their telemetry indicates that fetching OCSP results is an important cause of slowness in the first TLS handshake. Firefox, they write, is today the only major browser still fetching OCSP by default for DV certificates, meaning the lowliest common form, not even OV, which is Organizational Validation, but just Domain Validation. It's like, yes, you've proven to me that you have this domain.

They said: "Earlier we tried reducing the OCSP timeout to one second," meaning that they were going to wait less long for an answer, which would, if an OCSP result was slow to come, they wouldn't hang the whole connection. They said: "But that seems to have caused only a 2% improvement in 'SSL time until handshake finished' metric." So this bug, as they call it, although it's not really a bug, it's a feature, is to disable OCSP fetching for DV certs. "OCSP fetching should remain enabled," they write, "for EV certificates. And OCSP stapling will remain fully functional. We encourage everyone to use OCSP stapling."

So essentially they're just saying, you know, we're being hurt because we're providing - we're being hurt in comparative performance because we're the last browser to bother

actually checking to see whether a DV certificate may have been revoked. And so we're going to stop doing that. Which is sad. And they are explaining that stapling they will still honor, if a server staples. And of course, as we just covered, that doesn't provide you with a firm guarantee. But we're getting there. But what we need now is the server technologies to properly implement the OCSP stapling so that certificates can then safely add the OCSP Must Staple, and then we'll finally have a sane and workable solution for revocation. But it's difficult to get there.

BleepingComputer covered some interesting research. Where is - I didn't follow the report down. I've got the PDF: ethz.ch. I know them, but I can't - it's not coming to mind who they are. But a bunch of researchers have discovered that the latest generation, which are the MLC, the multilevel cell SSDs, can be subject to physical adjacency disturbances similar to what we've seen with DRAM and the Rowhammer attacks. And our listeners will remember that I deliberately purchased SLC, single-level cell SSDs for GRC just because I was suspicious of MLC. It seemed like kind of a cheesy thing to do in order to squeeze more bits into the same space. Yes, you get more density. But you're going to lower your integrity. And I don't need that much server space anyway.

Anyway, there are two attacks they've come up with, one called "program interference," a program interference attack, as they named it, occurring when data is deliberately written in specific ways and in MLC SSDs results in a 4.9 factor increase in the SSD's error rate. This, they write, can corrupt the SSD's stored data and shorten the device's lifetime. So this is bad. This is like a statically appliable attack to deliberately damage an SSD. The second attack they call "read disturb" occurs when a large number of specially patterned reads, which is very Rowhammer reminiscent, are performed within a short time. The researcher said these read disturb errors will "corrupt both pages already written to partially programmed word lines in memory and pages that have yet to be written, thus ruining the SSD's ability to store data in a reliable manner in the future."

Now, I would counter this, and I did not have time to look at their paper in depth, but it is uncommon that software probably has the kind of access that would be needed. It's not clear to me whether sector-level read and write access is granular enough to induce these kinds of problems. But I wanted to put it on our radar that it looks like it's not just DRAMs that are vulnerable as a consequence of this rush to density. Remember that it's the newer DRAM, the DDR3s and 4s that, as a consequence of the crazy density that they now have, that minimize the cell sizes in DRAM that subjected them to this noise induction from adjacent row interference.

Similarly, it's the crazy rush to density which caused manufacturers for competitive pressure reasons to need to put multiple bits of data into the same cell by creating multi levels of charge. And then they need to discriminate among those levels of charge in order to re-extract the bits. And so there are now two bits and even three bits per cell. And that just reduces the error margin. And then of course it makes them vulnerable to, now we know, this kind of attack. And this kind of attack looks like a persistent attack, that is, a static problem, where Rowhammer is just a dynamic problem.

Also it came to my attention that Twitter has just discontinued their support of the Do Not Track browser preference. They wrote: "While we had hoped that our support for Do Not Track would spur industry adoption, an industry standard approach to Do Not Track did not materialize. We now offer more granular privacy controls, and you can learn more about them outlined in the articles below." And I looked, and they're completely, you know, I mean, like it's nothing.

And so what I think this really means, reading between the lines, is that Twitter has chosen to no longer respect the user's browser sending of the DNT, the Do Not Track

query header. They've chosen to ignore it. And it's difficult to see how, if they were truly concerned about respecting privacy, they would choose to disable a privacy-enhancing feature that they had already implemented. I mean, they had already done the work. So why decide, oh, well, we changed our mind, we're going to take it out?

So what must actually be happening is that they want to be able to track users whose browsers are sending out the Do Not Track beacon, and this statement is their way of justifying that. They're essentially saying "Other services are ignoring their visitors' explicit requests to not be tracked, so now we're going to ignore it also." Like, okay. Yeah. Unfortunate.

I got a bunch of tweets, not surprisingly, from people telling me, informing me that the LastPass Authenticator would now be able to back up our two-factor data online. Robert tweeted: "@SGgrc @LastPass Authenticator can now back up your two-factor data online." Samuel Movi said: "@SGgrc I guess you won't be using your QR printouts much. In the future LastPass will store two-factor codes alongside your passwords." And Sean Stephens tweeted: "@SGgrc Someone @LastPass is listening to Security Now!. LastPass Authenticator now backs up to LastPass account for recovery and sync."

My response: This is not a good thing. This is bad. Multifactor security requires heterogeneous deployment.

Leo: Yes.

Steve: The idea is that my password and my time-based tokens must both be provided to log on. But if the same service contains both my password and my master set of time-based token keys, a single compromise of that service renders my entire use of a second factor meaningless. Also, I don't want an authenticator that is willing to export my time-based token keys. Anyone who gains control of a device's UI could export the key set. So the optimal solution is what I am doing and what I have recommended: Choose an authenticator app, not LastPass's app - and I don't like LastPass's app either because the things are so huge you've got to scroll through in order to see a six-digit number. I'd much rather have - I'm using Google's Authenticator, although there's another one that I haven't gotten around to looking at that apparently has an even denser display, which I would prefer.

So choose an authenticator app that deliberately does not allow for time-based token key export, like Google's Authenticator does not. And you take responsibility for managing the time-based token keys yourself. Yes, it's more work. But it's also the only way to actually obtain the security offered by a second factor. That's not something you want to relegate, and certainly not to the person who's storing all of your first factors.

Leo: It's why I never use the LastPass Authenticator.

Steve: I will be printing out my QR codes. I have a growing sheaf of printed out QR codes. And when I set up a new one, I just go through, it takes about a minute to say, here, sniff this, sniff this, sniff this, and then I'm good to go. And it's a one-way process. It makes much more sense to do it that way.

Okay. So yesterday was the 44th anniversary of the invention of the Ethernet. And as I said, Bob Metcalfe tweeted that "Ethernet's 44th birthday is Monday. I sketched this" -

that is, the diagram on the first page of our show notes - "to include in a typed memo," he tweeted, "on May 22, 1973 at the Xerox Palo Alto Research Center." And so I just wanted to note, to give Bob some major props because the Ethernet was a truly brilliant innovation. At the time, solutions involved some sort of centralized controller to negotiate access. There really it was no good definition of a network. We were just sort of - we were pre-network.

And remember that the traditional mainframe and terminal model was that all the terminals would connect back to sort of a concentrator device, and so they would all have their own connections. But this whole concept of an inter-workstation network was new. Now, IBM had something known as the Token Ring. And the idea there was that you would literally have a ring topology where workstations were connected, and the so-called "token" was like a baton. And so a workstation would hand the baton to the next workstation. If it had something it wanted to say or send, it would send it. And then it would hand the baton to the next workstation. If a workstation received this baton, and it did have nothing to say, it would immediately forward it to the next one.

So this token would circulate around the network. And it was only the holder of the token could speak on the network. And so that sort of solved the contention problem, well, did solve the contention problem, but actually at a cost of great complexity because then what happens if a workstation hangs, as they used to back then, when it had the token? Now the token was lost, you know, like the baton being dropped. And so you needed some provision for token regeneration. You needed all sorts of systems in place to deal with what happens, you know, I'm not getting the token, who has the token.

Anyway, it really escalated. Bob's brilliance was to conceive of a network where all the peers, where everybody connected was a peer. And they were on a common backbone, a common ether. That is, so everybody could hear everybody else. And then he developed a brilliant and minimal simple logic. And that is, anybody that wants to speak listens, because everybody can hear, listens until nobody is speaking. Then they choose a random number which is how long to wait before speaking. And if, when that time is up, nobody is still speaking, they speak. And while they're speaking, they're also listening to see if anybody else happened by chance to start speaking at the exact same time because, before you speak, you make sure nobody is.

But there is a chance two people could both start at the same time. So if that happens, they will both hear the other, or all three of them, or all four of them, however many happen to try to speak at the same time. So they then choose another random number, which is how long to back off before retrying. And that's it. It allowed a simple logic to exist in every NIC, Network Interface Controller, and it solved the problem. Everybody got to talk. And they all managed locally when they talked with a simple set of rules that they would obey, and the system worked.

And that is the Ethernet we have today, whether it's a wireless Ethernet, which is WiFi, or it's a wired Ethernet, that system was so good that it won. And the only problem, in the same way that the Internet guys who developed packet switching realized, okay, this is not perfect, but it's good enough, that is, we're not going to guarantee packets. We're going to have this just sort of opportunistic routing, and on top of that will be a protocol that solves the problem that buffers may overflow, packets may get dropped, routers may go down and so forth. Similarly, there's a problem with Ethernet. And that is, as the amount of traffic on that particular Ethernet segment, that ether backbone, whether it's wireless or wired, as the amount of traffic increases, think about what happens. At some point everybody has too much to say. And so the incidence of collision, of packet collision, where two people are talking at the same time, increases.

And so the efficiency of Ethernet falls to the point of failing as you begin to approach the maximum, actually well before you approach the theoretical maximum bandwidth of the system. So whereas IBM's Token Ring actually allowed you to closely approach much more of the maximum capacity, it did that at the cost of great complexity. Bob's breakthrough was being willing to sacrifice getting every last bit of bandwidth out of the Ethernet in turn for vastly simpler operation. And that was 44 years ago yesterday, May 22nd in 1973, as we were graduating from high school, Leo, that Bob came up with this.

Leo: Yes, he did. And a few years later Esther Dyson got everybody together. I love these pictures. Is that next?

Steve: Yes, yes. A bunch of people tweeted them to me for a reason that you'll see.

Leo: Yeah.

Steve: There's a young Bill Gates there, second from the left.

Leo: Gary Kildall, John Sculley, I'm not sure who...

Steve: Gary Kildall. Yup, Sculley second from the right.

Leo: Is that - I'm wondering if that's Lee Felsenstein? I'm not sure. Guy in the Russian outfit is, not Mitch Kapor...

Steve: Bricklin. Right?

Leo: Dan Bricklin, that's right, yeah.

Steve: Dan Bricklin.

Leo: He was funny.

Steve: He dressed up that way, and Esther loved that because...

Leo: He was Davy Crockett, it looks like.

Steve: Well, because he was a pioneer. And so he was dressed up as a pioneer, the idea being that we were pioneering the PC industry.

Leo: This is Esther Dyson used to have an amazing conference, Release 1.0. And these are some pictures. There's Jobs and Sculley.

Steve: Look how young he was, how young Jobs was, and Sculley, yup.

Leo: This is fun. Who is that?

Steve: And there's Esther, and there's Bill in the background.

Leo: Bill Gates, yeah, very young also. And wait a minute, who's that young fellow? Who's that guy? Dark-haired, mustache.

Steve: Yup.

Leo: Steve Gibson. This is 1984; huh?

Steve: Yup. I was there with all those guys.

Leo: Were you there as a pioneer? What were you doing there?

Steve: I was on a panel, and I also was the keynote speaker on one of the mornings.

Leo: Nice.

Steve: So, yeah.

Leo: Mitch Kapor. Gary Kildall was there.

Steve: There's Mitch, yup.

Leo: These are Esther's pictures. This is a lot of fun. This brings back some memories, some of these people.

Steve: Yes, back in the day.

Leo: Back in the day.

Steve: So I have this week's groan-worthy riddle, and I apologize in advance: How did the WannaCry hackers get away?

Leo: I don't know, Steve. How did the WannaCry hackers get away?

Steve: They ransomware. Ooh. Sorry.

Leo: Okay.

Steve: Okay. So Richard Romick sent me a nice tweet. He said: "I have a quick SpinRite testimonial for you. Sorry if I make any Twitter faux pas, as this is the first time I have sent a message." And Richard, for what it's worth, you did a very nice job. He said: "It started when my WD My Book Live Duo NAS told me one of the drives were failing. While I don't have any reason to mistrust WD, I felt it was a conflict of interest when their device says I need to buy a new WD hard drive. The NAS requires a specific line of WD hard drives to operate normally," he added. "Therefore, I decided to SpinRite it. I grabbed an old ASUS desktop computer out of the closet, and happy to see that the motherboard reported SMART statistics to SpinRite. I ran a Level 2 SpinRite scan on it. The scan was all blue across the board, and the ECC statistics looked good, no red.

"However, whenever I stuck the hard drive back in the NAS and ran a long test, everything was good to go again. I'm guessing SpinRite performed some refreshing of the data it found hard to read, which the NAS had interpreted as the drive failing." I think that's exactly right, Richard. "Thanks for the great product. I'm now running both drives on Level 4 just to be even more sure." So Richard, thank you for the note.

And one one-liner from Barry Coggins. He wrote: "Laptop was running hot with loud fan speed. Ran SpinRite, and all is well and quiet again." And of course we've seen that often. One of the things that happens is that there's, like, people don't appreciate the amount of energy required to move the head. The head is accelerated and decelerated very fast. Well, that creates heat. And one of the things that people have noticed is SpinRite sometimes causes their laptop to overheat because it's having to move the head as much. I expect that we're going to see 6.1 doesn't do that because the heads should move much more smoothly and uniformly because it'll be reading 32MB blocks at a time, rather than 32KB blocks at a time. So a thousand times the amount of data per read, which the drives are optimized for and should make things run, not only much faster, but also cooler.

But in this case, if the drive was having a problem, it was having retries and rereads, causing its head to generate much more heat or the drive overall to consume much more power than it would otherwise. And running SpinRite fixed the read problems and cooled off the drive and allowed the laptop's fan to slow itself down and everything to run cooler, which of course we know is much healthier for computers. They don't like heat.

And a couple of closing-the-loop comments from our listeners. Someone whose handle is @mcepl said: "Hi, Steve. Let me react in longer form [so this was a DM] on SN-612 [last week's podcast]," he says, "and your reflection on blaming the victim. I'm afraid that the advice you gave was so simplistic as to be dangerous. Yes, I can imagine that some MRI machine, completely air-gapped, really can run whatever" - well, he said "archeological." Maybe he meant that. I thought he meant architectural, but maybe he means archeological - "system one chooses. I understand that you, with some level of

understanding of computer security, can run Windows XP on your machine; although, as I recall, even you are now on supported Windows 7, aren't you?" No, I have that machine next to me. I haven't switched to it yet because I haven't been willing to accept the downtime it would take to move. I will eventually.

"However, when we are talking about 90% of NHS computers, then I assume neither air-gapping nor sophisticated use of computer can be assumed there. However, my biggest problem with this whole affair that I have been shouting to anyone whom I could talk to is the use of general-purpose desktop-oriented operating systems for everything, including things that are clearly inappropriate, like arrival/departure monitors, Twitter status and so forth is crazy and should be stopped. Yes," he says, "I work for RHT." And that's Red Hat; right? "But I would never think that even RHEL would be a good idea to be used there. And it is way more modular, so it can be stripped down more. You don't even need a hard drive for such monitors, or they should be read-only. Just receive message via network and display it. Why do you need Internet Explorer, Notepad, and FreeCell for that?"

For what it's worth, I am somewhat sympathetic to this. I do think that it's unfortunate that we see Windows error dialogues on marquees in Vegas and in the terminals in airports and popped up on ATMs. Unfortunately, it's a function of the fact that it is very easy to develop for Windows because of the huge array of tools that are prevalent. People are learning to program Windows. You can program Windows in any language that ever existed. So that's what ends up getting used. But I 100% agree that there are probably much more appropriate and less expensive and vastly more secure operating systems available. But they would cost more to actually deploy and to get out in the field, so that's not what's being done.

Leo: Actually, there was a rumor, I don't know if it was true, that the National Health Service in England was thinking of moving to Ubuntu, which...

Steve: That would be great.

Leo: It wouldn't be perfect, but it'd be less of a target, I would guess, yeah.

Steve: Yup.

Leo: Than old Windows.

Steve: Bob Beaudoin, I think how I pronounce that. Sorry, Bob, if I mispronounce your name, Bob Beaudoin. He said: "Did you hang onto the minted bitcoins? Sure worth something now." And I did a little bit of follow-up. And of course that's correct. They went north of $2,000 per bitcoin over the weekend. And of course we've seen a 450% increase over the last couple years. It turns out - here's the math, which is somewhat distressing. Had any of us purchased $100 worth of bitcoins seven years ago, we would be sitting on $72.9 million today.

Leo: What? Whoa. That means Satoshi is rich.

Steve: Yes. Yes.

Leo: Wow. Wow.

Steve: Seven years ago $100 of bitcoin is equivalent now to just shy of $73 million. Back then, let's see, the price was 0.003 cents on May 22, 2010. Wow. So just...

Leo: That's amazing.

Steve: It is incredible. Bitcoin trading hit $2,185.89 in the early hours of Monday morning, yesterday, hitting a fresh record high after it first cruised through the $2,000 barrier over the weekend, according to CoinDesk data. Yeah, so I guess Mark Thompson, minting in his garage, is feeling pretty good right now. Wow.

Oh, and Kostas Kristilas said: "Steve, in show 612, also last week, you were talking about the Android O restructuring and its similarity to Windows NT's HAL. You had some difficulty remembering the third architecture supported. I'm pretty sure," and he's right, "it was DEC's Alpha." And remember I said Intel and MIPS, and then Alpha. And then he says: "In addition, later on, it supported PowerPC and Itanium, the latter two in Windows 2000." And then he says: "Later on, that was ARM."

ETC Maryland said: "@SGgrc Windows S is not better because it's closed. It is still Windows and will not be secure. Just because an ecosystem is closed doesn't equal secure." And he says: "If that was the case, then open source means less secure. It doesn't." And I would take issue with this. I disagree. I think that an operating system where the only applications that can be run on it are curated is absolutely more secure. Not perfect. We've seen examples where bad stuff gets through the Apple Store, or gets onto the Android, you know, the Google Play Store. But as soon as that's known, both of those closed ecosystems are able to respond. Contrast that to Windows, where anybody can and does download anything from anywhere. I mean, that is a formula for disaster.

So I don't know if he misunderstood, but I think he's completely wrong. I mean, you're trading off a huge amount of convenience, the fact that you can't download anything you want from anywhere. But I would argue what you are getting in return is dramatically more security. And so, yeah, it does turn it into a limited toy, as opposed to a radically open workstation. I choose the latter. But I think there's certainly a place for Windows S, for example, in the educational environment, where a browser and an editor, an office suite, just a few things is all you really need, and not the kitchen sink that we have with Windows. I'm not going to be putting my apps there because I'm not going to be developing for the UWP system. But it remains to be seen whether this succeeds or not, or whether Windows S is just another Windows Phone.

Somebody tweeted a question from 609: "How are all these people able to unlock others' iOS devices when, after a few incorrect attempts, iOS enforces the passcode and wipes the device?" And I tweeted to him, and I'll just share the answer here. The device must support some sort of "fail count" in nonvolatile memory. So the people hacking phones, like the Cellebrite folks that I assume he's talking about, take a snapshot of the nonvolatile memory before they make their first guess. They then make the maximum number of failed guesses. Then they restore the snapshot, thus resetting the fail count to zero, and then make another round of guesses. Then refresh the snapshot again, make another round of guesses. Essentially that allows them to reset the fail count each time.

And I'm absolutely sure Apple - and I haven't kept track of it, whether Apple already has implemented a fix for that, or whether it's coming. But if it's not already in, in the hardware, you know it'll be in the next generation of hardware.

@sassymanjohnson is this Twitter handle, asks a question from Security Now! this week, which is last week, he said: "How did folks get the source code to WannaCry?" Because we were talking about people, like the MalwareTech guy, finding this domain name in the source. Almost without exception, everybody is using a beautiful tool known as the IDA Disassembler. If you just google "IDA Disassembler," you'll find it. And it is a beautiful multiplatform tool for reverse-engineering the compiled machine code of these exploits. So you drop it in there, this thing performs a static analysis, finding jumps and breakpoints, identifying the tops of subroutines, breaking the code apart into all of the pieces, the individual pieces that originally formed it, and then showing how they interconnect and intercommunicate and link together. And, where possible, it finds regions of memory that looks like strings and shows them to you as a string of text rather than as a series of bytes.

So it does a whole lot of like the first level, first pass reverse-engineering. And then somebody who understands this stuff is able to go in and look at what the code does, add labels, and essentially further populate it with information from their own inspection. So IDA Assembler, that is the go-to tool for this kind of forensic reverse-engineering of code for which you do not have the source, you only grabbed a copy from a machine that got infected, and you want to know what it does and how it works.

And lastly, JC tweeted: "I keep asking why NTP" - the Network Time Protocol - "cannot be hacked. No response." I don't know if he's asking me, or if he's just asking the ether. But, he says, "We're at a level that in theory it could. If not, why not? Podcast, please."

And I'll just say, yes, it can be hacked. NTP is both a UDP and a TCP protocol, but typically it's carried over UDP in the same way that DNS is because it's much faster. So it's not encrypted. It could be hacked. Although it's not really clear what a hack of time would give you. Many things are, like we were talking about certificates, if you lied about the date or time enough, then the certificate would be declared invalid, either not yet valid or now expired. So that Kerberos technology is time-based, and there have been some hacks against it.

But anyway, it's certainly the case that you could play games with time. But we haven't run across - we have run across exploitable NTP servers where the NTP service itself can be exploited, and we've discussed those. But you could certainly see an NTP response going by carrying the number of milliseconds since January 1, 1970, and change that into something else, and the recipient would now believe it's the wrong time, but it's not clear that that gets you a huge advantage. So, yes, now you have a response. Can be hacked. Maybe not worth bothering.

Leo: Wasn't the, I mean, aren't NNTP servers used for reflection and amplification attacks? I mean, the server...

Steve: That, too yes.

Leo: I mean, the server itself isn't - has been hacked, yeah.

Steve: Yup. That, too.

Leo: Is that it?

Steve: We're done.

Leo: Nothing more you want to say?

Steve: 613.

Leo: In the can, as they say. We do this show every Tuesday about 1:30 Pacific, 4:30 Eastern, 20:30 UTC. If you want to come by on a Tuesday afternoon, it's a great way to spend the afternoon or evening, depending on where you are. You could also join us in the chatroom at irc.twit.tv. But Steve, you know, he lives at GRC.com. That's his website, the Gibson Research Corporation. Lots of stuff there, including, of course, SpinRite, the world's finest hard drive and maintenance utility. And also this show. You can get audio of the show and handwritten, carefully contrived transcripts by Elaine Farris at GRC.com. We have audio and video at our website, TWiT.tv/sn for Security Now!. And of course the best way is just, you know, subscribe. Find your favorite podcast application and search for Security Now! and subscribe because you want to collect the set. You want all 613. You really - I'm not kidding you. You really do.

Steve: That's right.

Leo: And they're all available at Steve's site and my site. And actually the feed itself only has the most recent 10 because otherwise the feed would be prohibitively large. You wouldn't want it downloaded every hour or so. We are a prime number, Steve, I'm being informed by Dr. Morbius in the chatroom.

Steve: Nice.

Leo: Yeah. One more of those and you could have a key, a public key, a private key. Thank you, Steve. We'll see you next week on Security Now!.

Steve: Thanks, my friend.