Security Politics

Description: This week we discuss the continuing Marcus Hutchins drama, the disclosure of a potentially important Apple secret, a super cool website and browser extension our listeners are going to appreciate, trouble with extension developers being targeted, a problem with the communication bus standard in every car, an important correction from ElcomSoft, two zero-days in Foxit's PDF products, lava lamps for entropy, the forthcoming iOS 11 Touch ID kill switch, very welcome Libsodium audit results, a mistake in AWS permissions, a refreshingly forthright security statement, a bit of errata, miscellany, and a few closing-the-loop bits from our terrific listeners.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-625.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-625-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here and, as always, whew, just in the nick of time. He's going to explain what that Apple Secure Enclave Processor exploit means and whether it's time to worry. He's also got more thoughts on Marcus's arrest and maybe why the feds are after him. There's a whole lot more. It's a jam-packed edition of Security Now!, coming up next.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 625, recorded Tuesday, August 22nd, 2017: Security Politics.

It's time for Security Now!, the show where we cover your privacy and security online with this cat right here, the man in charge of the operation, Mr. Steve Gibson of GRC.com. We were talking reverse Polish notation before the show today.

Steve Gibson: Ah, yes, our favorite calculator modes.

Leo: Well, did you see that calculator? It was Andy's pick for MacBreak Weekly. It was a perfect reproduction on an iPad of the HP-65. It was amazing.

Steve: Nice. I did not see it.

Leo: You didn't see it? Oh, you would love it. I bet you even have an HP around,

probably four in the freezer; right?

Steve: Yeah, I actually just, you know, here's two within reach at the moment. One is the 15C Scientific, and the other is the 16C. So they're never far away. And in fact just last week I changed one out because the "8" key was beginning to get a little funky, and I was noticing I was having some entry errors. So I thought, okay, I've got 12 more of these.

Leo: Wait a minute. You use it?

Steve: Oh, daily, yes. Remember, I'm an engineer, too, Leo.

Leo: But we have computers for that now; don't we?

Steve: No, nothing is [crosstalk].

Leo: So look at this. You will love this. This is a perfect reproduction of an HP-65. And what's really cool, you know all the cards that they came with.

Steve: Yeah, yeah, yeah, the little magnetic...

Leo: They're all available for free.

Steve: Oh, that's nice.

Leo: So the next time you have to do your capacitance of parallel plates calculations, you just load that card into the HP-65. Watch how they put the card in.

Steve: Oh, my lord. Nice.

Leo: And it reads it, and there's a little description of it. You've got your paper tape here. You can advance it. Even the - what's really nice, Andy was pointing this out, even the segments on the display, they're not just using some font. They're actually really drawing it.

Steve: Nice.

Leo: Pretty sweet, huh?

Steve: It's very cool.

Leo: I think it's one guy in Switzerland who makes this software. It's the HP-65 or rather RPN-65 Pro is what he calls it. [Crosstalk] HP.

Steve: Well, and we've talked - I have on every one of my iDevices a copy of PCalc. And I think it's available on Android.

Leo: Oh, I love PCalc, yeah.

Steve: I mean, it's - when I'm out and about, I don't carry a calculator with me. I'm beyond that stage. But that's actually absolutely my go-to calculator. But when I'm here and I've got - I can physically hold this thing in my mind, the HP is just - it's just perfect.

Leo: Yeah, that makes sense, yeah.

Steve: We started the podcast recording, didn't we, a while ago.

Leo: Oh, yeah. I'm recording. Yeah, why?

Steve: So this is No. 625 for August 22nd. And I was really struggling for a name for this because nothing really stood out. But I want to talk about a number of things swirling around Marcus Hutchins, who as we've discussed for the last couple weeks was picked up at the Las Vegas airport upon trying to leave to return to his homeland, his home country, Britain, by U.S. law enforcement. And there's an interesting technical side to this which caused me to call this podcast "Security Politics" because, as often happens, when we don't have enough facts, guesses fill the void. And there is some interesting stuff that I wanted to address about what is going on in this realm, so we're going to talk about that.

We've got the disclosure of a potentially important Apple secret that I wasn't able to watch you guys discuss on MacBreak because I was busy pulling all this together. So we'll have my take on that. A super cool website that our listeners are going to love that includes a browser extension. Trouble with Chrome extension developers being targeted by attackers. A problem with the communication bus standard in all of our cars, which cannot be fixed. An important correct from ElcomSoft regarding that benchmark we shared last week which was wrong in a very important and critical way. Two zero-days in Foxit's PDF products. Lava lamps actually being used by a company we know well for entropy.

Leo: Ah.

Steve: The forthcoming iOS 11 Touch ID kill switch. A very welcome Libsodium audit result from our friend Matthew Green. A mistake in AWS permissions and what that means. A refreshingly forthright security statement from a company, I don't even know

what they're making, but I just loved what they said in their "What could possibly go wrong?" It's so forthright and upfront. A bit of errata, some miscellany, and then - time permitting, and I didn't expect we'd have much, so I only pulled two - interesting closing-the-loop bits from our terrific listeners.

Leo: Nice.

Steve: So I think a great podcast. So, wait. They have their own top-level domain?

Leo: Yeah, AWS. You didn't notice that. All of the ads about AWS...

Steve: Yow. Well, there's...

Leo: Hey, it's only a couple hundred grand; right?

Steve: There's an expression of clout.

Leo: Yeah.

Steve: Holy...

Leo: No, you can buy it. No, you can buy it. When ICANN announced this - WordPress bought .blog. And it was like, I don't remember the exact price, but it was like $150,000.

Steve: Oh, wait a minute, where's my wallet? I need that.

Leo: You need a GRC, steve.grc.

Steve: Finally, a use for those bitcoins that I've been holding onto.

Leo: How many do you have? You said you got 50 in one blow.

Steve: Yeah. I woke up the next morning after the podcast, and there was 50.

Leo: Now, you couldn't do that now. In fact, all the big...

Steve: No.

Leo: Does Mark still do his bitcoin mining thing in Arizona?

Steve: Yes.

Leo: He does. Because the key to it now, and this is - by the way, Satoshi Nakamoto, very smart, the whole thing was planned almost as if he knew exactly what he was doing, to cost more over time in terms of resources, particularly power.

Steve: We discussed it in our bitcoin episode [SN-287] years ago that there was a deliberate tracking, pre-planned, exponential curve where the difficulty of the coin would follow a trajectory. And it automatically scaled independent of the rate at which hardware became faster because it used time. And so if there was like a sudden breakthrough in the ability to solve these problems. And the problems are coming up with an input to a hash that has some number of zero bits. And the more zeroes in the output, the more difficult it is to come up with something - you don't care about the non-zeroes, but you just have to come up with something that generates, has some number of zeroes like at the right-hand side of the hash.

And so, for example, if it was just you needed one zero, well, half of the things you put in would end up with hashing to a zero in the least significant bit. If you required two zeroes in the two least significant bits, one quarter of the guesses would immediately give you just random guesses of what to put into the hash function would give you two zeroes in the least significant place, and so on. So by slowly increasing the number of zeroes that are required, this scales the difficulty of coming up with a value which hashes with that many zeroes and number of least significant bits. And so the algorithm is independent of the rate of change of hashing ability. I mean, it was immaculately conceived.

Leo: It's brilliant, yeah.

Steve: And we covered all this back in our bitcoin podcast [SN-287]. So if anyone's interested, it's all laid out in a podcast years ago.

Leo: But the upshot of it is that it's not merely better hardware, and there's custom ASICs and stuff now. Really it's the cost of power.

Steve: Yes.

Leo: So the reason Mark can do it, and it makes economic sense, I guess, is his power must be really cheap. But most...

Steve: Correct. In California it is no longer economic...

Leo: You can't do it.

Steve: No, you cannot mint bitcoin.

Leo: No, power's too expensive.

Steve: Apparently there's a massive facility under Niagara Falls.

Leo: Yeah, because they have hydro, yeah.

Steve: Exactly.

Leo: But most miners are now in China, and I suspect subsidized by the Chinese government. And they're mostly near hydroelectric plants. And I think it only makes sense because they're paying virtually nothing for their power. Anyway, you have what, 50?

Steve: Yes. Back in the day my one i7 machine, left running overnight - and back then that's how many - so that's how many bitcoin you got when you solved one problem.

Leo: Wow.

Steve: And so I woke up, and it said, oh, 50 bitcoin. It's like, oh. And, you know, that wasn't a big deal back then. Now we're at $4,000 for a bitcoin.

Leo: And I read in Forbes earlier today that one guy thinks it's going to go to something like $600,000 per coin. So you, sir, have a retirement plan.

Steve: Yeah, I've got to find those. I have them somewhere.

Leo: Do not cash in those bitcoins. In fact, it's worth enough to buy .steve, if you wanted it.

Steve: Oh, don't tempt me.

Leo: You could have 50 bitcoin or .steve.

Steve: That's the ultimate vanity domain.

Leo: OMG.

Steve: *.steve.

Leo: Oh, man. I'd like .leo. I really would like .leo, man.

Steve: Oh. Well, you know, maybe if you just brush your tongue some more, Leo...

Leo: There's probably some other hoops to jump through.

Steve: ...you can convince your wife to give you permission to...

Leo: Yeah. "Honey, can I buy a TLD?" "I don't know what that is, but if you need it...." "Yeah, I do."

Steve: Ooh, well...

Leo: Apparently from 2012 the price of an application for your own TLD is $185,000.

Steve: But it is not that much.

Leo: It's not going up at the rate Bitcoin's going up, that's for sure.

Steve: That's certainly low, Leo. That's, oh, boy. Okay. So the caption on this photo reads - our Photo of the Week. Quote: "Every employee has been implanted with biometric access RFID tags enabling secure access control to our extremely security-sensitive facility. It is utterly state of the art and cannot be penetrated."

Leo: And there's the picture.

Steve: The lesson, of course, being, ah, yes, the human factor can defeat any form of security that has been designed.

Leo: So somebody's put a big rock keeping the door open.

Steve: Yes. Meanwhile they're all limping around with, like, bandages on their thumbs because they've been implanted, and they're waiting for this thing to heal. And someone walks over and says, oh, screw this, puts a rock out to hold the door open. It's like, okay, yeah.

Leo: Well, now, doesn't Level 3 do like one of those airlock things? You've got double doors. You can't have somebody butt-riding you.

Steve: Yeah. I was told it's all computer controlled, camera monitored. And you have automated access. But the guy told me something I will never forget. He says: "When you leave, make absolutely sure this door is closed because our system monitors it, and all hell will break loose if this door remains open for longer than is reasonable" for someone to walk through carrying, you know, maybe two people at each end of a big server kind of a thing. So it's not allowed to remain open for long. And there are people on-premise. There's a security guard who will come running with his hand on his hip in order to make sure that nothing nefarious is going on. So, yeah.

Speaking of nefarious or not, Marcus Hutchins. One distressing piece of information which came to light that we haven't covered here is that Marcus, who as we know pled not guilty, is facing six charges and up to 40 years in prison. So I'm sure all of us feel as I do. We need justice to be done, and I'm so glad that, as I have said each week, that one thing we can be assured of is that this has come to the attention of people who can provide him with a first-class defense so that there will be no miscarriage. And for that I'm glad.

We also learned that the U.K. knew what was going to happen. They were aware that Marcus - who, as we know, helped them. He stopped the WannaCry worm which was decimating the U.K.'s entire NHS, their National Health Service IT infrastructure. GCHQ knew that he was under investigation by the FBI before he traveled to America and that he would be walking into a trap that was being set for him before he was then arrested by U.S. authorities for these alleged cyber offenses. Multiple reporting indicates that the British government allowed him to wander into this trap because it saved them the headache of what would then have been a highly charged extradition proceeding with the U.S., who is of course an ally. So they just said, oh, have a nice trip.

Okay. So what's developing in this case? Any time something important is happening, and this is why I titled this "Security Politics," people will form opinions. And then those opinions evolve into positions. Egos become engaged, and those positions start being defended, often before or beyond the point where they're supported by fact. It's the politics of humanity. And this is what's starting to develop in this interesting case with Marcus because there's a lot of background, but it doesn't have the context that is required for us to know what it means.

So during the past week there have been a number of new reports, some suggesting that Marcus was more deeply involved in the development of the Kronos banking malware, others arguing that the code was lifted from him by the Kronos malware authors, and some even alleging that he was more directly involved with the creating of the WannaCry worm. So some of this confusion surrounds the origins and refinement of a commonly employed programming technique known as "function hooking" and the need for something called a "trampoline." These are well-established, well-understood terms of art for antiviral software and also employed by malware. Now, to create somewhat of a historical background, 2.5 years ago, on January 8th, 2015, Marcus at his MalwareTech.com blog posted a two-part tutorial, an explainer about these technologies, function hooking and trampoline.

So to give everyone a sort of a sense for who he is in this context, he wrote, 2.5 years ago, 2015: "A lot of my articles have been aimed at giving a high-level insight into malware for beginners, or those unfamiliar with specific concepts. Today I've decided to

start a new series designed to familiarize people with malware internals on a programming level. This will not be a tutorial aimed towards people creating sophisticated malware, but security enthusiasts looking to better understand it."

So then he has a topic, "Inline Hooking." He says: "What is it?" Okay. "Inline hooking is a method of intercepting calls to target functions, which is mainly used by antiviruses, sandboxes, and malware. The general idea is to redirect a function" - that is, a function call, a call to a function - "to our own" - he meant code - "so that we can perform processing before and/or after the function does its work. This could include checking parameters, shimming the function, logging calls to it, spoofing its returned data, and filtering calls. Rootkits tend to use hooks to modify data returned from system calls in order to hide their presence" - and we discussed rootkits years ago using exactly this technology.

And again, we discussed rootkits and this technology years before Marcus posted this. So this wasn't his invention, and he's not claiming that it was. He says: "...while security software uses them" - that is, hooks - "to prevent/monitor potentially malicious operations. The hooks are placed by directly modifying code within the target function, called 'inline modification,' usually by overwriting the first few bites with a jump instruction. This allows execution to be redirected before the function does any processing." And he concludes, or I'm concluding my quote of the top of his post: "Most hooking engines use a 32-bit relative jump which is opcode hex E9, which takes up five bytes of space." And then he goes into detail in his second part, the assembly language code, mixed C and assembly, of the actual implementation of this technology.

I looked at it this morning. I've never seen that code before. Yet I read it for the first time, and I was completely familiar with every aspect of it. So I don't know, from what Marcus subsequently tweeted that has gotten him in some trouble, I don't know - he may have felt that he invented it. It may be a refinement of something, I mean, of what had to be common practice 2.5 years ago. I mean, rootkits are older than Marcus is. So I don't quite understand why he then later said what he did. But, for example, I have code from 1988 which does this because I had a...

Leo: You might not want to say that out loud.

Steve: Well, I had a developer working for me named Michael Toutonghi, who was brilliant. And we were developing a super high-performance disk cache back in the DOS days and the early Windows days that we called "Propel." And Mike needed to hook the operating system and the DOS compression engine and all this stuff. And this is the way you do that. So Mike went on to Microsoft, where he became one of very few Distinguished Engineers, which is a formal designation of the top people they've ever had. And Mike was the original architecture of the entire .NET system. He did all that.

So my point is that, back in 1988, this is what we were doing. And this reference to the trampoline, if you have a function, and you want to hook it, you want to intercept some other call to it, well, you need to replace the beginning of it with a jump instruction to you so that, when something else tries to invoke that subroutine, instead of seeing the beginning, instead of encountering the beginning of the subroutine, it encounters the jump instruction you have stuck there.

So, but notice that, in putting a jump instruction there, you have had to overwrite the first few instructions - the first few bytes at least, five bytes, which is why he referenced the size of the 32-bit jump instruction - you've had to overwrite the original beginning of

the function. So what is done is there's an instruction interpreter as part of the hooking system which reads the instructions, like the beginning of the function you're going to hook, in order to extract an even number, that is, the exact number of bytes and instructions that can then be relocated to your own code.

So you first look at what's there, figure out what the instructions are that you're going to smash by putting a jump there. You copy those instructions to your hook. Now the jump that you put there jumps to what's called the "trampoline," because you're going to jump off of it again. So you jump onto the trampoline, which executes the beginning instructions that you had to overwrite to put your jump instruction there and then jumps back and continues execution. So what you've done is you've hooked that function. Anyone who calls that runs through you first before going in.

So you have two things you can do. You can call, after your hook gets control, you can call the function yourself, that is, you call your own little trampoline, which then invokes the function. And when it's done it comes back to you. That allows you to filter the result. Or you can inspect the parameters that are being used first and, for example, abort the function, just return to the caller, so don't do what it said. For example, there's a function called "virtual protect" which turns on protection for virtual memory. Well, you could just short-circuit that so software thinks it's calling the virtual protect function, but it's neutered. It doesn't happen.

So the point is this is all super well understood. Everybody, you know, for decades has been doing this. But then, oddly, a month later, Marcus tweeted something. And his tweet contains the "F" bomb, and you know me, I don't use it casually, and I avoid it on this podcast. But I decided, and for a while I had concatenated it or hyphenated it, and I thought, it just doesn't convey his sense. And it's important for us to understand who he is.

So I'm going to read his tweet. I have a link to it in the show notes. On February 7th - so remember, his posting was January 8th, 2015. A month later, on February 7th, he posted, and it's still there on Twitter: "Just found the hooking engine I made for my blog in a malware sample. This is why we can't have nice things [bleep]."

So he's not happy. And it's clear he's not happy. Maybe it was an exact copy of what he did. But, I mean, it's not like they couldn't have gotten it elsewhere. It's not like this wasn't like a well-known, well-established technique. It was. And one of the things that's confused people is his use of a particular instruction.

Dan Goodin, writing for Ars Technica, said: "Shortly after his arrest in Las Vegas two weeks ago, the tweet resurfaced" - that is, this tweet from 2.5 years ago - "and almost immediately it generated speculation that the malware Hutchins was referring to was Kronos. An analysis of Kronos" - and, by the way, that's the well-known banking malware which is a known source of trouble. "An analysis of Kronos soon showed that one portion used an instruction that was identical to the one included in the code Hutchins published in January 2015."

Okay. Now, the instruction in question is what's called a "monotonic function." It exists in the Intel instruction set. It's a compare-and-exchange instruction. There are separate compare instructions, and there are exchange instructions where you swap two values. But performing those two functions separately, that is, doing a compare and then a conditional exchange, that allows the possibility that a thread context switch could occur, that is, an interrupt could occur.

Back in the old days, when we had a single core, you have a single processing core, but

you've got a multiuser, multitasking, or more properly a multithreading environment. What that means is that many things are going on at once, but you only have one actual CPU. So typically there's a hardware timer that is ticking in the background. And when that ticks, it yanks control away from wherever the processor was back to the OS, which then gives another thread some time to run. So this gives the illusion that all the threads are running at once, where in fact we're just switching among them very quickly.

Well, there are so-called "race conditions" that you can get into where, for example, you have something that needs to remain coherent in a multithreaded environment. Say, for example, you want to increment a 64-bit value, but you've only got a 32-bit chip. Well, the math is not hard. We know that you increment the low 32 bits. And if the carry overflows, that is, if it wraps around to zero, then that means you need to increment the high 32 bits to create a 64-bit counter.

But you cannot do that. It's hard for people to grasp. But you cannot do that safely in a multithreaded environment because there is a chance that you could increment the low side of that counter. And before your code has had the chance to immediately execute the conditional increment of the high 32 bits, it's interrupted, and another thread comes along and increments the low side. Well, now, if the first thread caused a wrap, then it didn't have a chance to finish yet. So now this other thread increments the low side, but it won't wrap.

Anyway, you can see what happens is this breaks a counter that was intended to reliably count the number of times it happened. But because it's not monotonic, because there's no way to do a 64-bit increment at once, we run into a problem. So in the old days, we would turn interrupts off. Before doing that, you would disable the hardware interrupts so nothing could interrupt your code in just that tiny interval of sensitivity where you must not be interrupted to create, to sort of fake a monotonic operation. Then you'd immediately reenable hardware interrupts.

But now we've got multiple cores. And so we don't have virtual multithreading, where we're switching around between threads. We have hardware multithreading, where actual physical cores are all working at once. So what had to be designed was instructions that could solve this problem. Thus the compare-and-exchange is one of the - there are several of these Intel instructions that are explicitly and deliberately thread safe, meaning that it does everything it needs to do at once in a single instruction, and you don't have to worry about it being, like, having to do a compare-and-test and a conditional exchange, which might get fouled up if thread-changing occurred. My point of all this is that this is the way you solve that problem. So Marcus's use of this instruction is like saying, gee, what instruction would I use if I had to add two numbers? Uh, how about the add instruction?

Leo: I'm sure a jury will understand that. That's just...

Steve: Yeah, I mean, and this is the problem. This is why I stopped being an expert witness is that these technical things are so clear to everyone who understands them. But I told you the story, Leo, about the judge who was on oxygen. I mean, I'm not kidding, he had a green oxygen tank next to him. I was an expert witness for NEC because Princeton Graphic had sued them over an ad about the MultiSync monitor being able to have a long life because it would adapt to whatever you plugged it into, which was true. But Princeton Graphic didn't have that technology, and they were annoyed. And so I was trying to explain to this judge who literally was on oxygen about, okay...

Leo: At least he was awake.

Steve: No, actually, he was nodding off, too. So maybe he needed to turn it up a little higher. But so this is the problem where, I mean, our listeners, you and I, I mean, we get this. But again, and this is why we need a technically competent defense with lots of charts and graphs who can explain that this instruction is - there's nothing magic about it. It has a purpose for which Marcus used it. And arguably there is no other way to solve that particular problem than to use it. And so the fact that some other code also used it absolutely doesn't mean that they got the idea from him or that he gave it to them.

I mean, first of all, remember, this was a public posting, a blog posting. So a month later he tweeted he was annoyed that apparently - and my sense is maybe he was claiming a little more ownership of this than he should. As I said, this is the way I would have solved the problem if someone said, okay, Steve, write some code, Windows code that does this. I'd just sit down, and I would write what Marcus wrote. That's what an engineer who understood the problem and what tools were available would use.

So anyway, again, exactly as you put your finger on it, Leo, the problem is the truth of this rests in some details which I'm just hoping a very good defense will be able to bring to light because that would be important.

Leo: Well, is that specifically what he's being accused of is that?

Steve: No.

Leo: We don't know.

Steve: Yeah. We don't yet know what the allegations are against him. Well, six counts of something with apparently up to 40 years of prison time. So that just can't be allowed. And, you know, as I was reading what he wrote, I'm thinking, no wonder he's popular. He's so literate, too. I mean, that was beautifully written, his description of the way the hooking and trampoline works.

Okay. So Part 2 is - that's Kronos. So what of WannaCry? There's a head of a security firm Immunity named Dave Aitel. He had a different immediate response to the reports of Marcus being, quote, and I'm just paraphrasing, "the hero of WannaCry," which of course we all understand because he created that domain, he registered that domain and shut down the propagation.

So in a posting, in Dave Aitel's posting at the Immunity blog earlier this month, he wrote: "But let me float my and others' initial feeling when MalwareTech got arrested: The kill switch story" - and here I will use an abbreviation because it's not important - "was clearly BS. What I think happened is that MalwareTech had something to do with WannaCry, and he knew about the kill switch. And when WannaCry started getting huge and causing massive amounts of damage, say to the NHS of his own country, he freaked out and, quote, 'found the kill switch,' unquote."

Leo: Ah. That's interesting.

Steve: Yeah, I know. "This is why he was so upset to be outed by the media." And then Dave says: "Being afraid to take the limelight is not a typical white hat behavior, to say the least." And then Dave continues, and backs off a little bit or, like, adds a little more depth. He says: "That said, we need to acknowledge the strategic impact law enforcement operations as a whole have on national security cyber capabilities, and how the lighter and friendlier approach of many European nations avoids the issues we have here in the states."

He writes: "Pretty much every infosec professional knows people who have been indicted for computer crimes by now. And in most cases the prosecution has operated in what is essentially an unfair, merciless way, even for very minor crimes. This has massive strategic implications when you consider that the U.S. Secret Service and FBI often compete with Mandiant for the handling of computer intrusions, and the people making the decisions about which information to share with law enforcement have an extremely negative opinion of it. In other words," he writes, "law enforcement needs to treat hacker cases as if" - and I like this - "they are the LAPD prosecuting a famous actor in Hollywood. Or at least that's the smartest thing to do strategically, and something the U.S. does a lot worse than many of our allies."

So we're all aware, I mean, addressing David's point, we're all aware of the very real concern, to draw an analogy, over a hopefully fictional, highly virulent super virus being developed purely for study in a lab, somehow later escaping from the lab into the wild and wreaking havoc upon humanity. Now, imagine a malware author who recently learned of the disclosed weaponized EternalBlue technology which we believe was weaponized and developed by the NSA, and who has a deep technical background in malware operation, being unable to resist the temptation of experimenting with that technology, and who creates an instance of a highly virulent worm which carries cryptomalware - why not - as its payload. It makes it a little more exciting and a little more real.

And because this researcher is not insane, he or she builds in a kill switch, just in case. Then somehow, some way, it finds its way out, like it's scanning. And within the researcher's network there was an unknown or unappreciated vulnerability. But, like, it gets loose, as viruses and worms will, and escapes its containment, exploding onto the public Internet. In this scenario, this wasn't what the researcher ever intended, but it happened nevertheless.

So now what does he do? He discovers his own responsibly installed kill switch and immediately registers the domain to shut down this creation of his which escaped his control. I certainly hope that's not what happened because then we're faced with a moral dilemma while, as Dave noted above, U.S. and global law enforcement won't have any dilemma. They will throw the book at Marcus.

So anyway, this is what's going on on the Internet in the technical forums and people combing through the evidence, looking at the code, and also just using, as Dave did, sort of his gut feel of how reasonable is the story that, wow, wasn't that great, you know, Marcus happened to find this wacky domain and thought, oh, I wonder what that does, and it shut down. I mean, remember how skeptical we were about why any actual malware author who wanted to be wreaking havoc with cryptomalware would put a kill switch in. It's like, why? It seems antithetical to the intent of that worm, except if it was an experiment. I mean, if someone responsible, fundamentally responsible did this in the

lab and knew enough to give it a kill, like just in case, and the worst happened. I mean, again, I think eventually we'll have answers to some of this really interesting case which I think is fascinating.

Leo: Now, if he wrote it, whatever his intentions are, I think he's in trouble.

Steve: Precisely my point. That's right. U.S. law enforcement will have no moral dilemma.

Leo: Yeah. Well, it reminds me of the Morris worm, remember. Robert Tappan Morris, when he wrote that - I think that was his name - when he wrote that worm, didn't write it to be a worm - a it was the very first computer worm - and was a little horrified that it was so effective. Didn't stop anybody from prosecuting him.

Steve: Okay. So I did not have a chance to hear you guys talk about the loss of or the escape of the Secure Enclave key. So I'll explain my take on it, and then I'd like you to add what you guys discussed on MacBreak.

Leo: I'll echo Rene's thoughts on it, yeah.

Steve: So, okay. Apple had been keeping a secret. It's unclear so far how important keeping that secret had been, but it is secret no more. We do know that cryptographic security does require some secret keeping. And as our listeners, our longtime listeners will know because we've discussed this in the past, the breakthrough, which was made quite some time ago in cryptographic maturity, occurred when we switched from keeping algorithms secret to developing keyed algorithms where the algorithms themselves could be made public, and then keeping specific instances of their usage keys were what was secret.

And when you think about it, just that, switching from secret algorithms to public algorithms and secret keys, that dramatically improves security because it distributed and spread the secret exposure risk. Beforehand, if the secret algorithm were discovered, every usage of it would simultaneously be compromised. But now, with keyed crypto, the disclosure of one key only compromises the secrets that are being kept under that key. No one else's use of the same algorithm under different secret keys is compromised.

So the point is there still remains secrets we need to keep, and understanding that sets us up for understanding maybe what this means. So Apple somehow lost control of their secret, that is, what they had been keeping a secret, the Secure Enclave firmware decryption key, and last Thursday it was published. That key allows for the decryption and exploration of the Secure Enclave Processor's firmware in full detail. And that was a closely guarded secret.

And so this is something that Apple has historically tried to prevent, not necessarily because it represents the end of mankind as we know it, but just because it's proprietary Apple code, and it's no one else's business. And it could be, emphasis on "could," an unexpected treasure trove for curious security researchers and malicious hackers alike. And the research community will doubtless be gaining much more information from it, like from the code that used to be secret that runs the Secure Enclave.

So I began this with a reminder about the nature of modern-day crypto. As with all modern cryptographic systems, disclosure of the algorithm is not a huge concern. So that's what we have had now is a disclosure of the Secure Enclave algorithm. So, per se, it's not a huge concern. But there are two dangers, the least bad of which is perhaps the more likely given the general history of complex unaudited code, would be that combing through Apple's code with an adversarial posture, which as we've often discussed is extremely difficult for Apple's own developers to ever really hold, thus the benefit of third-party audits, those with nefarious intent might discover some mistakes in the Secure Enclave Processor's code that could be leveraged into a powerful exploit that Apple could do nothing to prevent. We don't know. But now there's a chance that a mistake could be found.

But the biggest danger, and I haven't looked at it closely enough, and it'll take a while for what this means to mature more and to emerge, the biggest danger would be that the Secure Enclave Processor firmware itself contains embedded private keys whose security is crucial and whose disclosure, for some period of time until Apple can arrange to change them or do whatever needs to be done, could represent theoretically a catastrophic information leak. At this point we don't know.

Hackaday had a nice write-up. They said: "The SEP [Secure Enclave Processor] is a security coprocessor introduced with the iPhone 5s, which is when Touch ID was introduced. It's a black box that we're not supposed to know anything about." But this hacker who goes by the handle "Xerub," "XErub," I guess, or "Zeerub," don't know how you want to pronounce it.

Leo: I said "Cherub."

Steve: Okay, yeah, that's probably good. Xerub has now pulled back the curtain on the Secure Enclave Processor. "The Secure Enclave handles the processing of fingerprint data from the Touch ID sensor and determines if it's a match or not" - that would be fun to look at. Apple doesn't want us to, but still - "while it also enables access for purchases for the user," and many other functions, of course. "The SEP is a gatekeeper which prevents the main processor from needing to access sensitive data. The processor sends the data, which can only be read by the SEP, which is authenticated by a session key generated from the device's shared key," blah blah blah. I mean, those details don't really matter. There's been lots of write-ups about it. Apple has given us a broad brush, and there was a "Demystifying the Secure Enclave Processor" talk at Black Hat not long ago.

And here's what was more worrisome. It also runs its own OS called SEPOS, Secure Enclave Processor OS, which has a kernel, services, drivers, and apps. And, yes, that frightens me because that means it's not just a simple crypto engine, it's a much more complex subsystem, the code for which is now public. And as we know, security and complexity are always in constant conflict. So the SEP performs secure services for the rest of the system on a chip that is the backbone of the iOS devices, and much more.

So this Xerub guy said, quote, after disclosing something that Apple desperately did not want to have disclosed and had deliberately kept secret: "Hopefully Apple will work harder now that they can't hide SEP, resulting in improved security for users." Uh-huh. Yeah, that's his concern or his hope.

What remains unknown is how this Xerub guy obtained this presumably large key. Now, I have a guess. My guess is that it's the same old problem we've discussed here over and

over and over. Some secrets can be kept, like a remote web server's private key, because the work it does is done remotely, and the only access we have is over a protocol like TCP, and specifically TLS, where the key is used, but is never - we have no remote access to it. And for it to do its job we don't need access to it.

But other secrets inherently cannot be kept, like, as we've often discussed, a DVD's secret key, which must be present and in use to play back an encrypted DVD. Or in this case, similarly, the Secure Enclave's firmware decryption key must also be at least temporarily transiently present to decrypt the processor's stored firmware in order for it to be executed and used. So the key is in there, and it had to briefly come out of hiding. And this guy, I'll give him props for cleverness, apparently managed to capture it somehow. So what were you guys talking about on MacBreak, and what were Rene's thoughts?

Leo: Pretty much like as yours were. One is that, of course, security through obscurity is never a great idea. However, it doesn't hurt to have a number of layers. And Apple's smart enough, he felt, not to have depended solely on that. Obviously it doesn't want people to bang on the SEP if they can avoid that. But now that people will be, my thought was Apple at this point needs to raise the value of its bug bounty because it's not really competitive. It's offering hundreds of thousands of dollars when malicious actors and governments are offering millions of dollars. And certainly, I mean, I don't know, but if there were an exploit in the Secure Enclave Processor, and it were to be sold to the NSA or a bad guy, that would be - probably a bad guy wouldn't have much use for it, I would think. But certainly a government might.

Steve: You think Apple has the money, Leo?

Leo: [Laughter] Nah, they're broke. So it's not a good thing, but on the other hand it can't possibly be the only security they've got for this thing. They just have other things going on.

Steve: Yeah.

Leo: And so, yeah, I mean, what's going to happen now, and it's happening as we speak, I'm sure, is all sorts of people are banging on this thing, seeing if they can find flaws. As we've learned in the show, all software has flaws.

Steve: And the more complex it is, the more likely those are.

Leo: Right. What wasn't clear to me is what, and I guess it really depends on the exploit, but what could be done with it. I doubt you'd have a mass exploit that would exfiltrate everybody's password. What good would that be anyway? It's really going to be one of those targeted, I would expect, vulnerabilities that somebody like the NSA could use if they got a terrorist's phone. Well, hey the Secure Enclave, the fingerprint, Touch ID is not protecting them anymore because we can get right to it. Right?

Steve: Yeah.

Leo: It could be used to unlock a phone, I would think.

Steve: The problem is, I mean, I was distressed to learn that it's so complex, that it's an operating system.

Leo: Yeah, that's interesting, isn't it, yeah.

Steve: Processors and drivers and blah blah blah. So, I mean, Apple has no need to publish it, so they chose not to. I don't think this is the end of the world. But we're not going to really know until the community looks at it. I mean, again, it could be that nothing will be found. It could be that there are some edge cases that can be leveraged. We just don't know. It's a functional module. They didn't need to share it. And they recognized better not to because it's complex. And so you'd rather not let people poke at it. Now, as you said, that's going to start happening. And we'll know more a month from now. And presumably this is all remotely, over-the-air updatable. And so Apple will change the key, revise the...

Leo: I wonder.

Steve: Yeah, I do, too.

Leo: It could be in hardware.

Steve: Yeah.

Leo: I wonder. And if you do change the key - I don't know. I wonder. Yeah, that would be the best-case scenario; right? Although people already have the SEP code, right, because the cracked one's out there.

Steve: It's already escaped.

Leo: I'm sure it's on GitHub.

Steve: It is, actually, on GitHub. That's where it is.

Leo: So [chatroom visitor] had an interesting thought. It would be possible to craft a malicious application that would, because you'd have access to Touch ID, you'd be able to authenticate purchases, for instance, using Touch ID. Maybe some faux authentication scheme so you could have the application buying stuff and you

authenticating it, without actually authenticating it. I don't know. It's hard to think of what exploits are possible. But no doubt some are.

Steve: Yeah. Well, and from a props to hackers standpoint...

Leo: Yeah, nice job, yeah.

Steve: Well, yeah, that. But also, what comes next? Apple has been a little, I don't want to say "snobby," but a little, you know, we know better than everyone. We're all about security and so forth. So a takedown of Apple is something I could see some hackers thinking, oh, I'm going to get them.

Leo: They've made themselves a target, yeah.

Steve: Right, exactly. Okay. I've got the nice, the coolest site and browser extension that a chunk of our listeners are going to love. We're all familiar with the common term TL;DR, you know, Too Long; Didn't Read. So this is a great play on that, TOS;DR - Terms of Service; Didn't Read. The site is TOSDR.org. The page's subhead reads: "'I have read and agree to the terms' is the biggest lie on the web. We aim to fix that." Isn't that great?

They said: "We are a user rights initiative to rate and label website terms and privacy policies from very good Class A to very bad Class E. Terms of service," they write, "are often too long to read." Often? Often? Yeah. "But it's important to understand what's in them." Okay. Maybe. I'm guilty as everybody else is of lying, fine, do whatever you're going to do. "Your rights online depend on them. We hope that our ratings can help you get informed about your rights. Do not hesitate to click on a service below to have more details."

And then they add: "You can also get ratings directly in your browser by installing our web browser add-on. This extension" - which, by the way, is available for Chrome, Firefox, Safari, Opera, and on its way for IE. They didn't mention Edge. "This extension will add a small icon to the right side of your address bar. If you are browsing a website that is not in our database, nothing appears. Click on the icon to get the summary of the terms provided by TOSDR. When you land on a website with very bad terms, a small notification will pop up in the bottom right-hand corner to warn you."

So, okay. So you can go without installing anything anywhere, just scroll down through TOSDR.org because right there on the home page is a bunch of samples, all of the sites that we all know very well - Google, YouTube, SoundCloud, GitHub, and so forth. And as is always the case, a grade is kind of arbitrary. That is, they're looking for specific features. They've got thumbs up, thumbs down, plus and minus things, and then they have to somehow end up, you know, distilling that into a simple linear scale. But they do break it all down so you can see if you care about the things that the site does or doesn't do and so forth.

But anyway, just a cool - this is nice to see, sort of a maturation of the industry. And maybe this puts a little pressure back on the companies who are, because no one is reading their terms of service, able to have egregious things in there that they're never

held to. But if it's summarized like this, and some light is shined on it, maybe that makes them think, okay, maybe we shouldn't have listened to our attorneys when they gave us this boilerplate that no one reads.

Leo: Yeah. You know, I'm not sure about the grade. I think that's maybe a little reductionist. But I love it that you can kind of look at this summary of terms.

Steve: Yes. You can scan through, oh, they do this, and they don't do that. And they do do that, and they don't do that, yeah.

Leo: That's great, yeah.

Steve: Yeah, that's neat.

Leo: And you can decide, well, do you mind that Zing doesn't allow a pseudonym? Is that an issue for you or not? Right?

Steve: Precisely. Precisely. Or that LastPass is there, and it's like, they reserve the right to change the terms and conditions without notice. Well, okay.

Leo: [Crosstalk] every single site. That's a thumbs-down almost everywhere. It's rare that you don't see that.

Steve: Yes. Well, and again, all of these big companies, they've got attorneys. And the attorneys, you know, are the ones who write this impenetrable boilerplate.

Leo: Yeah, that's boilerplate. They're going to put that in, no matter what.

Steve: Yup. Yeah, anyway, very cool, TOSDR.org. So speaking of browser extensions, this was another one of these sort of foreseeable things. As we know, Chrome is now the number one browser on the Internet; and, if you have enough RAM to keep it happy, deservedly so. And except for the fact that it doesn't give me tabs that I want, someday I will be using it. But I'm still happy with Firefox. Developers of Chrome extensions may have weaker security than Google. So attack the point of least resistance, which is less cautious developers.

"At the beginning of the month we had the news of the hijacking of an OCR add-on called Copyfish. By 'hijacking' we mean that attackers managed to break into the systems of the Copyfish developers and compromise their code base so that the now 'quasi-legitimate' extension" - that is, it's carrying some additional baggage that the developers didn't intend, a legitimate extension downloaded by innocent Chrome users that could then attack them in various ways.

Now, according to an update more recent at Proofpoint, Copyfish has been joined by seven additional legitimate Chrome extensions which attackers have taken over and used

to manipulate Internet traffic and web-based ads. There's an extension called Web Developer, one called Chrometana, Infinity New Tab, Web Paint, and Social Fixer, those five. And then they also believe that two, the TouchVPN and the Betternet VPN, were also compromised in the same way at the end of last month, at the end of June. Oh, no, I'm sorry, month before last, end of June.

So Proofpoint said: "At the end of July and beginning of August, several Chrome extensions were compromised after their authors' Google account credentials were stolen via a phishing scheme." So what's happening is extension developers have become a target because, if their security can be compromised, then their publication of extensions to Chrome can be infiltrated with malware.

And browser extensions are powerful add-ons to browsers, whereas JavaScript code downloaded ad hoc from almost any website today, as we know, is inherently untrusted and is therefore executed with extreme caution and containment. Browser add-ons, while not fully trusted, have significantly greater conditional trust and so can perform many actions, on behalf of their users and autonomously, which are explicitly denied to page-based, you know, web page-delivered and -based code. So this represents a rational attack vector and a means of compromising high numbers of unwitting Chrome users, and we're seeing this.

So I don't know what the takeaway for our listeners is. But, you know, it's very much like how it's not completely free to download every Android app you ever encounter. I mean, there's some wisdom associated with, do I really need this? Because there's, you know, the probability of anyone of them hurting you is very small. But the more of those probabilities you have add up, then the total probability becomes a bit greater. So I would say maybe just don't go crazy.

And who knows, maybe Google will be able to come up with some solution because they certainly are very proactive about their browser security, and they're not going to want well-intending add-on developers to be the delivery vector of problems for their own Chrome browser users. And it's a problem because the reason you do extensions is to bind tightly across sites to the browser and to obtain those extra capabilities, the things you can't do on a page. But with that capability comes responsibility.

Okay. So a group of researchers has developed an attack on the CAN bus. The CAN bus, C-A-N, stands for the Controller Area Network. That is the glue that is what makes all cars for almost two decades run. I mean, once upon a time there were wiring harnesses that ran through our cars. And you had a harness going to all of the lights in the back of the car. Now you don't. Now you have power going to the lights in the back. And in the lights is a processor on the CAN bus. It's exactly analogous to - you can think of it as like the old-school coax original Ethernet where there was a network backbone, and nodes tapped into it. Well, that's how our cars are now. There's a bus that carries this CAN protocol, and everything hooks into it - your side view mirrors, your door locks, your power windows, the engine itself, the entertainment system, the backup lights, the antilock brakes, I mean, it is the glue.

So CAN messages, including errors, are called "frames." And these guys write: "Our attack focuses on how CAN handles errors. Errors arise when a device reads values that do not correspond to the original expected value of a frame. When a device detects such an event" - that is, when it receives an erroneous frame - "it writes an error message onto the bus in order to 'recall' the errant frame and notify the other devices to entirely ignore the recalled frame."

So it's very Ethernet-like. Remember that something that's transmitting on the Ethernet

is receiving its own transmission in order to detect a collision with another simultaneously sent packet. So although the reason may not be collision, if something trying to emit a frame onto the CAN bus detects a failure to do that successfully, it then says, whoops, and sends this recall message saying ignore that, what I just said. And everybody will accept that.

It turns out that this occurring is very common, and it's usually due, they write, to natural causes, a transient malfunction, or simply by too many systems and modules trying to send frames through the CAN at the same time. So again, exactly analogous to Ethernet. It was the packet collision problem. And we've talked about how, as a consequence of that, Ethernet does fabulously well up to a point. But you can't ever get, like, saturation of the Ethernet backbone because the probability of collision then rises fast, and everybody spends all their time retransmitting collided packets.

So, and I'll just note, too, that as an engineer, cars are one of the most horrifically noisy, electrically noisy environments you will ever find. I mean, so the system needs to be extremely - any network communication needs to be extremely robust in the face of static surges and spikes and engine noise and just all kinds of events that can happen.

They write: "If a device sends out too many errors" - okay, that is, so something is erroneously transmitting mistaken packets and recalling them - "as the CAN standard dictates, it goes into what's called a Bus Off state, where it's cut off from the CAN bus and prevented from reading and/or writing any data onto the CAN." So it's sort of a self-healing feature so that if something, like if a taillight processor goes wonky and starts flooding the bus with nonsense, there's a system that is able to take it off the bus. So this feature is helpful in clearly isolating malfunctioning devices and stopping them from triggering other modules and systems on the bus because this all, of course, is being used for communication, communication among different systems in the car.

This is a feature that these guys' attack abuses. Their attack triggers this feature by deliberately inducing sufficient errors that a targeted device or subsystem on the CAN is forced off the bus into that Bus Off state and thus rendered inert and inoperable. They have demonstrated that this in turn can drastically affect the car's performance to the point that it becomes dangerous and even fatal, especially when essential systems like the airbag system or the antilock braking system are deactivated. All it takes is a specially crafted attacking device introduced to the car's CAN through local access. And I'll note that we've also seen remote CAN bus access and the reuse of frames already circulating in the CAN, rather than injecting new ones. So there were previous attacks of this sort, but they've further refined it and made it worse.

So how did we get here? Well, this is another classic Security Now! lesson. The original system is old. And it was designed just to work in the face of the assumption of mutually cooperating interlinked systems. And as we have seen so often of things like this, it was never designed to be secure against deliberate attack because, back when it was designed, there was no way for an attacker to gain remote access to the CAN bus. But as we know, our newfangled autos, with all their fancy new connected-to-everything-else features, create that opportunity. We've on this podcast talked about how things like inserting a CD into the entertainment system or cell phone WiFi or OnStar radios can be used to access the car and its bus at a distance.

So the CAN bus dates back to 1983 at a company named Robert Bosch, where it was first created. The protocol was officially released in '86 at the Society of Automotive Engineers, the SAE conference in Detroit, Michigan. The first CAN controller chips were introduced by Intel and Philips, and they appeared on the market a year later, in '87. And the 1988 BMW 8 series was the first production vehicle to feature a CAN-based multiplex

wiring system. So no more crazy wiring harness. We're just going to give everything power and communication network. And today virtually all cars are on - they are based on this CAN bus backbone.

So the problem is we can't fix this. Maybe in some future, I mean, maybe there's a way to revise the spec in a backward-compatible fashion so that, over time, as new CAN-attached components are created, there can be some agreement about bumping this up. But the problem is this system we have wasn't designed to be attack proof, just like the original Internet was not designed to be attack proof. Thus it isn't. And it'll be interesting to see if there's a way to evolve this spec eventually so that there is a way to make it robust, since now automotive security is a thing, that is, security of its network that glues all of this together is a thing. And it never used to be.

I'm going to jump ahead, and then we will take a break. But I wanted to share a nice piece of email from someone whose domain I really got a kick out of. And I thought of you, Leo, because I know you. Jesse Harris has the domain coolestfamilyever.com. Which I just thought, that's just so neat. Anyway, I saw his email, actually it was sent to support, and Greg forwarded it to me. The subject was "SpinRite saves the day again." He said - and this was a really nice piece of mail. He said: "I know I can't get enough of hearing how great something I've done is and figure you're probably the same."

Leo: Awww.

Steve: "I purchased SpinRite around three months ago," he wrote, "when my primary desktop started experiencing lots of strange IO errors and related lags in the OS. I ran a Level 2 scan on my SSD" - so it was solid-state-based - "which turned up no issues. Despite this, it's been performing absolutely normal ever since." So there's the first instance. Again, and we've discussed this, just letting SpinRite run over your drive. That was a Level 2, fastest mode available, and it just fixed whatever was wrong. No errors were surfaced because it showed the SSD controller where its problems were, and so the controller said, ooh, shoot, and, like, fixed it. And so, yeah, those are all done down in the firmware of the controller so it doesn't represent an era that surfaces, but SpinRite helps the SSD to find and fix its own problems.

But then he wrote - so that was, what, I think it's three months ago. Now he said: "This morning I had the worst-case scenario of two drives failing in a four-drive RAID array." And he wrote this is a RAID-Z1 on FreeNAS. So that sounds like a ZFS. He's running my favorite file system, ZFS. So he had one drive redundancy; right? But, bang, two drives failed. He was in trouble despite having RAID. He says: "I pulled one of the drives, ran a Level 2 scan, and dropped it back into the box. One quick command later, the pool was up in a degraded but functional and accessible state as I scanned the other drive reporting errors." He says: "It's almost like the drive just needed to see that I did, indeed, mean business when it decided to whip itself back into shape."

And then he said: "For a long time I balked at the $89 price tag, and yet now I wonder if you're even charging enough for what this software is capable of doing. Looking forward to what's in store for the future of the 6.x series once SQRL is done. Jesse Harris." So Jesse, thanks for letting me share that with our listeners.

Leo: Nice.

Steve: So, okay. Big important correction.

Leo: Uh-oh.

Steve: For users of 1Password. Not our mistake, ElcomSoft's mistake. Last week we talked about the NVIDIA GTX 1080 state-of-the-art GPU accelerated brute-force testing of those four password managers: Dashlane, LastPass, Keepass, and 1Password. And what was notable there was that LastPass appeared to allow very few, that is, the fewest number of brute-force guesses per time interval, whatever it was, minute, second, whatever. 1Password, in the graph presented by ElcomSoft last week, was more than twice as fast, meaning twice as many guesses per second in that data. And then Dashlane and Keepass were also faster. And we remarked that Office 2016, of all things, was the slowest to allow brute forcing - well, except SQRL, that is, like, way on the other side of the decimal point. But still, these are in the multiple thousands of guesses per second.

Okay. Turns out they screwed up. Probably as a consequence of the 1Password guy saying, wait a minute, that's not right. It turns out that they had snapshots of old data, and they mistakenly used the old database data for the new GPU-enhanced benchmarking, which failed to take into account years of improvement in the algorithms. As a consequence, 1Password, rather than being twice as bad, if we say "bad" is enabling more guessing per second, it is now, when properly benchmarked, the best of all by default, even better than Office, which was previously the most laggardly, which is what you want in brute forcing, the most resistant to brute forcing.

Unfortunately, the two charts I have in the show notes have a different X axis scale. I don't know why they didn't make the chart the same because that makes it confusing. So I made a point in the show notes of "note also the change in X-axis scale." None of the other numbers changed from last week to this week except 1Password, which went from the erroneous 95,000 guesses, and I think it's per second, down to the proper number. And I didn't look into why they have two, but they show two. One is 6,200; the other is 3,000. But when you compare that with Office at 7,300, 1Password is more difficult to brute force than even Office 2016, that was by far harder than all the others.

So props to ElcomSoft for immediately correcting the record and fixing this, and I wanted to correct it for our listeners. 1Password, in its default settings, is way slower. And so, as I said last week, normally these things are tweakable with configuration settings. And people who want more security ought to consider slowing down their password-based key derivation function, if their password manager lets them change things, because in doing so they would make it more difficult to brute force.

At the same time, remember that that's only one class of attack. If something evil gets in your machine and watches you type the password in, it doesn't have to guess because you just gave it to it. So a class of attacks it's worth being brute-force resistant for.

Just a quick heads-up for users of Foxit, who produce and publish some PDF readers. We've spoken of them in days past. There were a pair of recently disclosed zero-day vulnerabilities that generated some headlines, but Foxit responded initially a little awkwardly. They said: "Our existing containment technology means these are not a problem. We're not going to bother with them." That apparently didn't go over very well anywhere. So they, "Oops, sorry, that's not what we meant. We meant that our containment will prevent anybody from being hurt until we get this fixed, and we're working on an improvement."

So for what it's worth, anybody who's using Foxit within the sound of this podcast, keep your eye out for updates and update yourself. You're not in danger right now because they do have a good sandboxing technology. But the vulnerability was maliciously embedded executables carried by PDFs could, if for some reason you had disabled the on-by-default protection, could then get loose in your machine. So again, nobody's in danger. But if you're using Foxit, it's worth fixing it.

And I ignored this story, this next one, the first time it came because I was like, okay, we've heard this before. But it's a big, well-known company, and they actually are doing this, and it's fun. There was sort of a question of whether this was just apocryphal back when we heard that Sun was using lava lamps as a source of entropy, the idea being that, when you look at the wax which is heating, and therefore being reduced in density compared to the oil that it's suspended in, and thus it rises up to the top where it cools off and then comes back down, the fluid dynamics are chaotic, inherently. You have no idea if it's going to break into three pieces, or stay as one big one, or exactly how it's going to undulate around in the oil that contains it.

And so it is, in the same way that, as I've spoken of before, a reverse bias diode junction that you put under stress by essentially breaking down its resistance to reverse electron flow, which allows electrons to tunnel in a way which is absolutely unpredictable at the quantum level, produces absolutely high-quality entropic noise. Similarly, looking at a lava lamp is entropy. And so our friends at Cloudflare, who now handle 10% of the Internet - and they're a sponsor; right, Leo? Cloudflare?

Leo: Not exactly.

Steve: No?

Leo: They were a sponsor.

Steve: Oh, okay. Well, anyway...

Leo: We like them. I love them.

Steve: Yeah. And they're handling 10% of the Internet's traffic. They actually have, and I have a picture in the show notes, a wall in the lobby of their San Francisco facility of lava lamps - different, actually alternating colors. There are some strange ones. I think that there's one that's just turned off. Sort of like, I don't know, red and green, kind of Christmas-y looking, lava lamps. That wall is being captured, the image of these banks of lava lamps is being captured at the rate of 1,000 times per second; 1 millisecond frame rate is being snapped.

That data is then hashed to produce non-algorithmically generated, utterly unpredictable, true random numbers. Remember that random numbers are only pseudo when an algorithm produces them because, if you know the state of the algorithm, then you are able to predict all of the future. Anything that is truly random, like a hash of the photo of this crazy lava lamp wall, that's truly random because this is going to produce, at a rate of 1,000 hashes per second, utterly unpredictable results. So it's just very cool. I mean,

it's simple, but it really works.

And so this is why I've been contending for some time that it is no longer the case generally for randomness to be hard to have. Early in this podcast we talked about the problems of generating entropy. And there are instances of devices, for example, if there was, like, no communication, so that you couldn't use the timing, the high-resolution timing of the arrival of wireless or wired packets, there's an example. Or if you have a wristwatch that has an accelerometer, it's producing entropy because you're - especially if it's on my wrist because I'm waving my hands around all the time while I'm doing the podcast. You know, it's spewing entropy. No one is going to be able to predict what the future is.

And of course you need to "whiten" it, that's the term for removing bias, and also make sure that things aren't stuck and that it's reliable and so forth. So there's post-processing you have to do. But anything that's got, like, a connection to the real world, the real world is normally producing unknown, unpredictable stuff. And if you just suck it all in and hash it, you get high-quality entropy, which is what I did in - I discussed it, We did a podcast called "Harvesting Entropy" [SN-456] because the first thing I wrote, the very first thing for SQRL was an entropy-harvesting technology to produce really good true random numbers, not pseudo.

A nice feature that will be of interest to our listeners, we've often talked about this question of fingerprint biometrics versus password. I guess there's news that maybe the iPhone 8 is going to use facial recognition in order to unlock? I've not been tracking the recent MacBreak Weekly podcast, Leo. Is that presumed to be happening on the iPhone 8, facial recognition?

Leo: Yeah, presumably. I mean, you know, these rumors.

Steve: Yeah, yeah.

Leo: The rumors are very strong they're going to replace, on the high-end newest iPhone, fingerprint Touch ID with a very snappy facial recognition that won't even have to aim directly at you.

Steve: Good. And I hope that it...

Leo: Or maybe it'll be eye recognition. I mean, that's not completely clear. Might be iris or something.

Steve: Yeah. I'm hoping that, if it's facial, and I'm sure they know this, that it needs to be stereo.

Leo: Yes.

Steve: They need to have cameras at...

Leo: There's many cameras on that thing.

Steve: Good.

Leo: Again, rumors. "Rumor has it."

Steve: Rumor has it. So another piece of information. This appeared in the most recent iOS 11 beta. And again, our listeners will like it because there's been the problem, I know that there are people who are not using Touch ID because they're worried about being compelled to put their thumb on the phone. And that worries them enough that they would rather enter a passcode than use the Touch ID to unlock. And as we know, the law is still completely confused, with judges coming down on both sides of both questions, of whether passwords are testimonial, whether someone can be compelled to unlock their phone through various means and so forth.

So Apple has added to iOS a new feature. If you click the power button, a.k.a. the sleep button, since as we know the phone always actually stays on, you click it five times in succession. That primes the phone to make a 911 emergency call and disables the Touch ID function. And it remains disabled, no matter what you do, until - very much like a cold power-up, where the only thing it will do is you must enter your device's passcode in order to enable Touch ID and unlock the phone.

So anyway, just a nice feature. It's not something that I'm worried about, but something to keep in mind if you're going, I don't know, through TSA. Or I saw somewhere called it the "cop escape" or something, where the cops are coming to get you, and so you quickly click your phone's power button five times, and now your fingerprint will no longer unlock. So anyway, it's nice that Apple's thinking of these things.

Leo: I think they were forced into it because in order to do a full screen without a bezel they had to eliminate the fingerprint reader. They wanted to do a fingerprint reader that would work through the glass, and apparently - this is all rumor - apparently they couldn't get it working in time. Why they didn't want to do a fingerprint reader on the back, I don't know, because that's what Google and everybody else does. But apparently they didn't want to. Probably the first [crosstalk].

Steve: Yeah, and for what it's worth, I can really see the trouble of, like, mixing it with the screen.

Leo: Right, that's hard.

Steve: Those are fundamentally incompatible. As we know, Touch ID is a high-resolution capacitive imaging sensor. And, boy, it does not want to operate in the presence of all of the noise of an LCD screen, like sharing the same space. I just - I don't know how you would do that.

Leo: Well, and if they could what they say they're going to do with the - I'm sorry. If they do what the rumor mill says they're going to do, this will be the best possible solution because, even when the phone's just - basically, all you have to do is have your face visible to the phone at any angle, and it knows it's you.

Steve: Wow. Super wide, it's a wide angle.

Leo: Yeah. So it could be lying on this desk, and you just sit there, and it says, oh, yeah, Leo's here. So it almost would be a - it sounds like, the way they're implementing, like presence detection.

Steve: I would call that "optical tethering."

Leo: Yeah. And that would be really cool because you basically would not have to do anything to authenticate.

Steve: Wow.

Leo: Right? Google's working on similar things, as we've talked about. Based on how you walk, your gait, how you talk, how you hold.

Steve: That's cool, too, because Jeff Arthur, who is the guy who's doing the iOS version, the iOS client for SQRL, he's got it enabled now with Touch ID. And this would mean that it would be, by tying the authentication of, yes, this is me wanting to use SQRL to log in, that just, I mean, it becomes even more magical. And it's already kind of freaky as it is. You'll remember a long time ago when I showed Jeff's client your web page through Skype of the QR code, and it logged you in over Skype, I mean, optically. So, yeah, that makes it even easier. I mean, even more magical.

Leo: Yeah, it's cool, yeah.

Steve: So the cryptographic toolkit which is the foundation of SQRL's crypto, which I chose three years ago, and which I recommend without hesitation, it's the state-of-the-art crypto, the elliptic curve crypto. It's designed to help people who aren't experts make no mistakes: Libsodium. It has been independently audited by a friend of the podcast, Dr. Matthew Green of, I'm looking here, Cryptography Engineering is in the coverage of this. The organization Private Internet Access released the results of its Libsodium audit.

They wrote that: "Libsodium is an open source cryptographic library that is used far and wide in projects such as Zcash, as well as internal applications at Private Internet Access. Private Internet Access is proud to have another audited tool in its software suite. The Libsodium security assessment was conducted by Dr. Matthew Green of Cryptography Engineering. Dr. Green previously completed the TrueCrypt audit with the Open Crypto Audit Project, as well as an OpenVPN 2.4 audit on PIA's behalf" - which we talked about before, or both of which we talked about before. "The assessment found no critical flaws

or vulnerabilities in the Libsodium library.

"Matt, who authored the Cryptography Engineering Libsodium Security Report, summarized the audit goals and results, saying: 'Over the past several months we conducted a detailed audit of the Libsodium cryptographic library at the request of Private Internet Access. Our goal in this effort was to help ensure the safety of an increasing number of applications that rely on Libsodium for cryptographic operations. We are pleased to report that our review did not uncover any critical flaws or vulnerabilities in the core library. Overall, we believe that Libsodium is a carefully implemented, secure cryptographic library.'" And you can tell Matt is a cryptographer because of that language. "Overall, we believe," that is, they've looked at it. They understand they're not perfect, either. But they're independent, and they looked at it trying to find problems, and they found none. So great news for Libsodium, not only for PIA's use, but for everybody else who's using it, including me.

There was a story about data stored at Amazon in an AWS cloud bucket. And I just wanted to note that this is - and so what happened was, sorry, the permissions on the bucket, which can be private or set to public, were accidentally made public, and the cloud-based content was available on the Internet. This is not, obviously, Amazon's fault. This is the fault of the user.

Leo: It's misconfiguration; right?

Steve: Exactly. It's a classic configuration error. But what was interesting to me is that, you know, this is the danger inherent in easy-to-use power, where what we're getting are sort of like toolkits, which are astonishingly powerful for their ease of use. But with that great power comes great responsibility. So just sort of a cautionary note that, yes, it's wonderful to have things in the cloud. It's wonderful to be able to have access to them anywhere. But it's not wonderful if you don't intend for everyone to have access, and they do. So permissions are important. And as you said, Leo, configuration matters.

Okay. There's this company. I don't even know what Turtl is, T-U-R-T-L.

Leo: I do. I use it.

Steve: Oh, good.

Leo: Turtle Hop?

Steve: Turtl, no, Turtlapp.com.

Leo: Yeah.

Steve: I was so impressed with their disclaimer. I've got the link in the show notes, for anyone who's interested. But I also have the entire thing on one page of the show notes. And, you know, completely open kimono, as I say, they say: "When is Turtl not [in italics] secure?" They said: "Here are some possible scenarios where Turtl's security

measures will fail you. We try to provide an exhaustive list so you're aware of the dangers of relying on Turtl." And I love it. The first thing they said under "When is Turtl not secure? When we make mistakes."

Leo: Good. I like it.

Steve: Bravo to them.

Leo: It's a large - it's not a commercial enterprise. It's a large open source project. So it's easier for them to be honest than a business.

Steve: True, true.

Leo: Right?

Steve: And so they said: "When we make mistakes." They said: "That's right, we're human. It's entirely possible that bugs in the Turtl client leave your data exposed, especially," they wrote, "at our early stage." And then I won't take everyone through its length, but they...

Leo: Yeah, maybe it is a business. Maybe they do intend to make it a business. But it's open source. I download it and run it on my server for no cost.

Steve: What is it?

Leo: It's basically - so when Evernote started charging and stuff, I looked at a bunch of different Evernote replacements. It's a Note app that you host.

Steve: I like it. I want one.

Leo: And then they have, well, what's nice is they've written really nice apps for the mobile platforms. So it really does give you a lot of the capabilities of Evernote at no cost to you. But you have to be able to run your own server. That's the...

Steve: I can do that.

Leo: Yeah, I thought you might.

Steve: So other examples of when it's not secure is, they say, okay, "When we make mistakes," as I said, but also "when you use a bad password." Bravo for saying that, of course. "When you invite someone to a board over email." And then they go into some

depth of essentially they use a shared secret technology where, if that got loose, that would be a problem. "When someone you shared data with is compromised. When you have malware installed." Right. I mean, yeah, that's correct. As we keep saying, if something infects your client, all bets are off. "When your operating system is compromised. When your hardware is compromised." And, finally, "When someone is holding a gun to your head." So, yes, they covered all the bases. And as we know, excessive coercion is sometimes the path of least resistance, unfortunately, for security.

Two pieces of errata. Delphi Ote said: "Road sign classifier misstatement." And this is from my discussion of that learning neural network training. He caught me in a mistake. He said: "You said it had 91% accuracy on data used to train it." He said: "That would be training set, not test set." And he's completely right. I completely missed that and skipped over it. The way neural nets are trained is with two separate sets that are disjoint. They're non-overlapping. And I said last week that it couldn't recognize the same stuff it was trained on. Wrong. They had a separate training set which is different from what it was tested on later. And so its recognition accuracy was 91% on the testing set which it had never seen before. So thank you, Delphi, for helping me correct that.

And, secondly, Emett Speer caught me in a subtler mistake. He said: "You misquoted the availability of S3." And I was having fun with it, the number of nines, but there actually are a gazillion of them. The 99.999999999, I think I got the number right, percent is the data durability. S3's availability is 99.99. So only four nines on availability, but a gazillion nines on - and, frankly, I'd rather, if you ask me, I can wait for my data. I just don't want it to go away forever. So I care more about the 10 nines of durability than I do - well, for my particular application. I'm not saying everybody. But for me, I'm just using it as a backup archive. So if I have to wait a minute, fine, if it blinks off briefly. But it never has.

Okay. And the Tweet of the Week, Leo, you'll get a kick out of this. And our listeners, it's a perfect tech nerd Tweet of the Week. Andy, who tweets from @remixman, tweeted: "LOL, @SGgrc." He's retweeting something from a Jim Birch, who said: "The sun is currently undergoing a denial of service attack. We hope to have full service restored soon." And of course that was from Eclipse Day yesterday.

And I was speaking in the last couple weeks, actually two weeks ago, about the availability of the third book of the third trilogy of Richard Phillips' Rho Agenda work through the end of August. So here we are on the 22nd. So is August a 31-day month, Leo? I think it is.

Leo: Yes, it is. Don't you know the old doggerel, "30 days hath September?"

Steve: Yeah, I count the knuckles.

Leo: I have to go through it each freaking time. Oh, you use the knuckle method. Ah.

Steve: Actually, I can sort of remember. I remember August 31st happening before, so it's probably going to happen again.

Leo: Yeah, yeah.

Steve: So anyway, through the end of August, all previous eight of the nine books, which I love and which many of our listeners have loved, back when I was discovering them and telling people about them, I was getting a lot of positive feedback. These are the teenagers that discover the second ship and get changed. And, oh, it's fun. Anyway, sorry, 99 cents each. So I created a bit.ly link for this, for this show: bit.ly/rhoagenda, R-H-O-A-G-E-N-D-A. I've also got the full link that that expands to. And what that link takes you to is an Amazon page where you can buy, for less than a dollar, well, a penny less, 99 cents each, all of the earlier eight books. And they are all fun. I mean, it's terrific. So if you thought maybe that sounds like a good idea, but you were waiting, through the end of the month - which we now know is August 31st - you can purchase them for 99 cents each.

Jan asked: "@SGgrc Could you talk about how to encrypt your storage on Amazon S3 for next week's episode of Security Now!? Or has it been done already?" And I'll just say again that I am a fan of CloudBerry Lab. I looked at their crypto years ago, talked about it here. I've looked at it recently. And it's the tool that I've chosen for GRC and for my own use. As I mentioned before, Sue is still using our original Jungle Disk account to back up her machine. CloudBerry Lab is a full-featured suite. So that's what I use. And, boy, as I mentioned before, they support every cloud provider you ever heard of and about three times that many that you've never heard of. So really good coverage.

And then, finally, on the question of the teapot, Leo, the HTTP 418 code.

Leo: Yes.

Steve: Remember that the 400 series, 4XX, are errors.

Leo: Right.

Steve: So the famous one is 404, and it's that Page Not Found. 418 was the I'm a Teapot error. Well, a lot of listeners knew more detail. I did, too, but I didn't want to go into it last week because as it is we never even got to any of our closing-the-loop feedback because we ran out of time. But there's a site called Sites Done Right that blogged back in 2013 in March: "What is 418 I'm a Teapot status code error?" And they said: "If you look through" - and there's some fun history here that I just wanted to share, and then that's the end of our podcast.

"If you look through the full list of HTTP status codes, you'll see one that really stands out: '418 I'm a Teapot.' So what's that all about? I'll explain in a bit," this writer says. "But first, a really quick introduction to status codes. Whenever a page or file is accessed on your site, whether it's a user accessing it in a browser or a search engine crawling a page, your server" - so he's taking the server end view. We look at the client end view, typically, where it's our browser fetching from a remote server receives a status code telling it how that HTTP fetch went.

So he writes: "Your server returns an HTTP status code in response to the request, and it provides information about the status. For example, some popular codes are '301 Page Moved Permanently' and," as he notes, "the ever-ubiquitous '404 Page Not Found.'

"So what's the '418 I'm a Teapot' all about? Well, the group of people who make these codes and set standards is the IETF, the Internet Engineering Task Force. To propose

new standards, the members release RFCs, or Requests for Comments, to the community. Every year, since 1989, they release a few humorous RFCs for April Fools Day; and, on April 1st, 1998, RFC 2324 introduced the 'Hyper Text Coffee Pot Control Protocol,' HTCPCP v1.0."

Leo: I love it.

Steve: Yes, the Hyper Text Coffee Pot Control Protocol.

Leo: And then that goes back to that Cambridge Coffee Pot. I guarantee you; right?

Steve: And it was fully developed, Leo.

Leo: Really.

Steve: Because these guys aren't messing around.

Leo: These are real engineers here, yeah.

Steve: Oh, yeah. So, he writes: "This was a brand new protocol for controlling, monitoring, and diagnosing coffee pots. Now, the RFC is pretty funny with lines like 'Coffee pots heat water using electronic mechanisms, so there's no fire. Thus, no firewalls are necessary."

Leo: Oh, how funny.

Steve: "If you have never read it, it's definitely worth a read," he says. "I added a link in the description below. Now, you may be asking, if this is a Coffee protocol, why the Teapot code?"

Leo: Yes.

Steve: "This is answered in Section 2.3.2 in that, quote, 'Any attempt to brew coffee with a teapot should result in the HTTP error code 418 I'm a Teapot, and the resulting entity body may be short and stout.'"

Leo: Oh, dear. Oh, dear. Oh, dear.

Steve: It's pretty bad, Leo. I mean, it'll hurt you. "And just like that, the '418 I'm a Teapot' code was born."

Leo: Was born, yes.

Steve: "Since then, it's been used in all sorts of wacky ways. Google even referred to 418 as a different error in their 2013 April Fools Joke 'Google Nose,' saying that '418 Scent Transfer Protocol Error indicates system congestion. Please try again later.'" So, yes. A little fun from the crazy RFC authors.

Leo: Really funny. "Halt and Catch Fire" is back. I forgot. You like it or don't like it?

Steve: I fell off of that wagon. I got a little bit into Season 3. It's like, okay, Cameron's hot, but it's like not worth it.

Leo: That's why I stopped watching from the very first episode, because they went there right away, and I thought, this isn't going to be good. Although I just noted Woz tweeted or Facebooked or something that it's his favorite show.

Steve: Wow. Well, I also abandoned "Twin Peaks" finally. It's like, oh, this is just...

Leo: It makes no sense, yeah.

Steve: Oh, it's a fever dream. I mean, it is just...

Leo: Yeah, that's a good way to describe it, yeah.

Steve: It's just like, okay, Lynch, what have you been smoking, and change brands because, boy. Oh.

Leo: Yeah. On the other hand, "Game of Thrones" is getting better and better; isn't it.

Steve: Leo, it's, oh, yeah. In fact, you know what's sad? And I was glad to see this. I encountered this somewhere in a link as I was researching the show. Some site was talking about how it's unfortunate that now they've sped it up so much.

Leo: Right.

Steve: It's like they're now in a hurry. I mean, it was...

Leo: Right, they have to...

Steve: ...painfully slow.

Leo: They have to bring a million threads together; right?

Steve: Oh. And the dialogue as those guys are walking through the ice was just...

Leo: Isn't that funny?

Steve: It was just delightful. But, boy. And now, you know, we're already through with our little half season. Next week is the end of this chunk. Do you know when the next one is? Do we have to wait till...

Leo: I think we have to wait. I don't think they begin shooting, or they've just begun shooting, so it's going to be another, yeah, year or something. I thought they were going to do half and half, like they'd shot the whole season and spread it out.

Steve: Spread it out, yeah.

Leo: But I don't think that's the case. I don't know. Some fan will tell us. But the thing that's interesting to me is this is the payoff for your devotion. I mean, they always said it's going to be a 73-hour movie.

Steve: Yeah.

Leo: And you've watched 60 hours.

Steve: My nephew, my sister's son and his wife are - I don't know how I happened to run across it. We keep in text touch, or touch text, I don't know, something. Anyway, so I just happened to mention it because, I mean, they've been so good, I've been trying to tell people who I think would be interested. And he said they never saw it. They're now on, like, Season 3 just a couple weeks ago. So maybe they're up to 4 or 5 now. And I just said to Evan, I said, it's worth it. I mean, binging is probably so much better than the way we've had to do it over the course of seven years.

Leo: Probably, yeah.

Steve: Which is like, oh, who was that again? Who? What? What? What? Because, I mean, you know, you needed a guide just because it was so stretched out.

Leo: What other TV show in the world would bring back a character you haven't seen in five years and expect you to know him?

Steve: Yes.

Leo: Right?

Steve: Yes.

Leo: Only GoT, baby. Only GoT. Yeah. I don't know if I'll have the energy, stamina. I'm going to have to wait until I'm in the nursing home and don't have anything else to do. But I would love to rewatch it. You know, Lisa's only seen, like, a bit of it. So I keep saying, you know, if you want to start at the beginning and go through the whole thing, I'm willing to try it. But 73 hours.

Steve: I know. I'm looking for someone who hasn't seen "Breaking Bad" yet because that would be fun to share.

Leo: Same here. And I've been thinking of rewatching that whole thing.

Steve: Yeah.

Leo: But again, you're right, it'd be more fun if somebody else - because that was probably the best TV series ever was "Breaking Bad."

Steve: I think it was, Leo. I mean, I really do. It's top of the list.

Leo: ScooterX says the "Game of Thrones" next season won't start production until October. So it's, yeah, it'll be next fall.

Steve: I guess they do have to wait for winter somewhere, don't they.

Leo: Winter is coming. I think winter is here. Steve Gibson, he's at GRC.com. That's where you can go to get, oh, so many great things. Everything's free there except one. And that's SpinRite, the world's best hard drive maintenance and recovery utility. That's his bread and butter. So do him a favor. Do us a favor. Do yourself and your hard drives a favor. Get a copy. GRC.com. You can do all the other stuff free - ShieldsUP! to test your router and your firewall. You can Shoot the Messenger and DCOMbobulator and Trouble in Paradise. I'm going way back now. You can also, of course, find out more about SQRL. Somebody in the chatroom is saying it's not clear from the website that it's a server-side technology. I mean, it's both. It's client and server.

Steve: Leo, at this point nothing is clear from the website. It is a mess. And so the plan is the installer technology is nailed. We're now happily tracking down little remaining things like should this text be red or green, and the focus doesn't work on 64-bit Win10 if

you don't use IE and Edge. You have to click the dialogue. I think I fixed that, but I haven't looked yet because I've been working on the podcast since I posted what I hope is a fix for that. So, and I have a bunch of other - I have a whole to-do list of little debris to clean up. But, I mean, it's, like, it's very close.

And what I'll do is, once it really seems done, I've built a FreeBSD server for the forums because I'm going to be running public forums to host forums for all the clients and developers, so there's a public place for this interchange to occur. So I'll get the client finished. And the problem is unfortunately I built it using an older x86 server, not realizing that ZFS and enough RAM needed an x86 platform. I've got one just sitting there, but I built it on the wrong platform. So I'm going to rebuild it, move everything I already did. The forums exist, believe it or not. They're there. I've just kept them offline until I'm ready.

So I'll stage the forums, we'll get that tested, and then we'll release the client publicly. The demo sites, there's multiple demo sites. There's iOS and Android clients and my Windows client. There will be then a place for people to ask questions and a community to solve problems and so forth. And while that's happening, then I go back and catch the website up that I haven't looked at since the beginning, so that it then represents a current statement of the spec.

The core pages I've been keeping current. But all of the intro background stuff I just thought, okay. For a while I kept rewriting it over and over. And I thought, okay, just stop. Let's get the technology finished, the clients running, everything going. And then I'll explain it all and describe it. So that's my planned sequence. And of course the second that's done it's back to SpinRite 6.1 and much more compatibility and a lot more speed, which everyone will enjoy. And a free upgrade for all users of 6.0.

Leo: Yay. So get over there, GRC.com. He also has the audio of the show and really nicely written transcripts, so you can read along. A lot of people like to do that. It makes it easier to understand for many of us. For some of us, we'll never understand. But you can also get audio and video on our site, TWiT.tv/sn.

We stream live, if you want to watch live and be in the chatroom. The chatroom is irc.twit.tv. The live stream is TWiT.tv/live, and we do that every Wednesday, right after MacBreak Weekly. Or, I'm sorry, Tuesday, right after MacBreak Weekly, roughly 1:30 Pacific, 4:30 Eastern, 20:30 UTC, if you want to come by and say hi. Otherwise, on-demand audio and video at Steve's site, my site, and, of course, in every podcatcher known to man. So no matter what you use, subscribe. That way, you know, you'd like a complete set. You want it all, Security Now!. Thank you, Steve. We'll see you next week.

Steve: Okay, my friend. Till then. Oh, and by the way, this is the first episode of Year 13.

Leo: Year 13. That's what I told the guy in the chatroom. You really needed to listen to all 13 years to understand what SQRL is.

Steve: And you'd better start now because, you know...

Leo: Seventy-three hours?

Steve: We're going to keep them coming.

Leo: That ain't nothing.

Steve: Make "Game of Thrones" look like a walk in the park.

Leo: I would guess the average length is probably an hour, although almost all the shows in the last few years have been two hours long. But we started really short. So you've got at least...

Steve: How could we do this in less than two hours? It's not possible.

Leo: There's probably a thousand hours' worth of show. Somebody knows, and they will tell us. Thank you, Steve.

Steve: Your caffeine-powered podcast.

Leo: We'll see you next time.

Steve: Bye, Leo.

Copyright (c) 2014 by Steve Gibson and Leo Laporte. SOME RIGHTS RESERVED

This work is licensed for the good of the Internet Community under the Creative Commons License v2.5. See the following Web page for details: http://creativecommons.org/licenses/by-nc-sa/2.5/