SECURITY NOW!

Transcript of Episode #582

Listener Feedback #241

Description: Leo and I discuss some serious concerns raised over compelled biometric authentication, then do a detailed dive into the recently completed audit of VeraCrypt, the successor to TrueCrypt. We've got more on web browsers fatiguing system main SSD storage and a bunch of interesting miscellany, including a question asked of Elon Musk: "Are we living within a simulated reality?" We conclude with 11 questions and observations from our terrific listeners.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-582.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-582-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. For the first time in a while we've got questions and answers. Steve's going to answer a whole wide-ranging list of topics. But before we do that, we've got security news, and he's going to talk about the VeraCrypt audit - very good news for people who want to find a replacement for TrueCrypt. Stay tuned. It's coming up next.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 582, recorded Tuesday, October 18th, 2016: Your questions, Steve's answers, #241.

It's time for Security Now!, the show where we cover the latest security news and help keep you safe and then explain how things work. And it's all done by this one fellow right here, the hardest working man in podcasting, Steve Gibson of the GRC company.

Steve Gibson: Yo, Leo.

Leo: Hey. Yo, Steve.

Steve: Great to be with you. So we don't have a lot of individual pieces of news. We're going to do a Q&A this week, just because there's a bunch of a fun stuff that our listeners have submitted, thoughts and observations and questions in some cases. But there were a couple significant stories. And the one that just burned up my Twitter feed was the announcement of the audit results from VeraCrypt, which of course is the successor to TrueCrypt.

Leo: I was thrilled that they audited it so quickly.

Steve: Yes, yes.

Leo: Because we went years without an audit for TrueCrypt.

Steve: Yes. And it's a consequence of the OSTIF, which is a small organization. And I can't remember what the acronym is for. But I have it in my notes. We'll get to it - Open Source Technology Improvement Fund.

Leo: Oh, I like it.

Steve: And so they generated the funding and then contracted with a French security firm who put two people on it. They spent 32 man-days - and I thought, okay, well that should be probably person-days - in order to go through it. But the depth of their analysis is really informative, as is what they found. There's also, that we'll talk about, some concerns, a new big concern about biometric authentication problems. And you missed a story that I had some follow-up for that I wanted to make sure you were aware of, where web browsers turn out to be fatiguing SSD main storage on systems. A ton of miscellany, including we need to address the question of whether we're living within a simulated reality.

Leo: Hah. How have we missed that one? That's a - yeah.

Steve: And then a Q&A. Well, because it happened since you and I talked last, with Elon going on the record saying, yeah, I think there's probably only maybe a one in billion chance that we are not in a simulated reality.

Leo: Yeah.

Steve: So anyway, it turns out that a lot of the popular press completely misunderstood what he meant because the only thing that we are aware of sort of popularly is "The Matrix," and he's not referring to that model at all.

Leo: Ah. Oh, good.

Steve: So I thought it would be fun to discuss that.

Leo: I'd love to.

Steve: So lots of fun stuff this week. So this is my second edition of the famous book that is often referred to as the "K&R text" I'm holding up in front of the camera for our listeners. This is the book, "The C Programming Language." And I looked around for my first edition copy, which is about half the thickness of this one - even so, very thin by any standards of modern programming languages.

Leo: It's funny, I have both the first and second, as well. And I keep the second here and the first at home.

Steve: So I've reread the preface so many times because I just get goose bumps. And I've referred to it on the podcast a few times. Just the first paragraph reads: "C is a general purpose programming language which features economy of expression, modern control flow and data structures, and a rich set of operators. C is not a very high-level language, nor a big one, and is not specialized to any particular area of application. But its absence of restrictions and its generality make it more convenient and effective for many tasks than supposedly more powerful languages. C was originally designed for and implemented on the Unix operating system on the DEC PDP-11 by Dennis Ritchie." And he passed away.

Leo: Five years ago.

Steve: What?

Leo: You know, I saw all this social...

Steve: What?

Leo: Yeah. He passed away the same week Steve Jobs did. No, you fell for it, and I almost fell for it, too.

Steve: How weird.

Leo: It was all over Twitter and Facebook that he'd passed away.

Steve: Yes, and I got sucked right into it.

Leo: But if you click the link, there's a date on it: 2011. But nobody looked at that. So, yeah, he passed away when Steve Jobs did. In fact, one of the reasons I remember...

Steve: And now that you say it, I remember.

Leo: Yeah, because people were saying, oh, everybody's making a big deal about Jobs dying. Nobody's saying anything about Ritchie passing away. Yeah. Anyway, but you know what? We deserve a tribute to Dennis Ritchie, five years later or a hundred years later. This was a great man and wrote, not just this, but much of Unix.

Steve: Yeah. He and Brian Kernighan.

Leo: Yup. K&R, they call this.

Steve: Yup.

Leo: And, you know, I actually haven't read the second edition. But the first edition, my first edition is well thumbed.

Steve: Oh, it's just - yeah. What I love about it is that it is so lean.

Leo: Yeah. It's just very simple.

Steve: Of course it appeals to me as Mr. Assembly Language, so...

Leo: It's as close to assembly language as you can get in a high-level language.

Steve: And many people shared with me a Photo of the Week, which I just got a kick out of. So this is the "you're doing it wrong" in a simple picture. And so this shows a password field which is highlighted in red because the system's not happy. And it's saying, "Password is used by another user."

Leo: Oh.

Steve: It's like, oh. So count the number of things that must be wrong with a system that is able to tell you that. Now, it's true that it could be checking the hash. And we know that hash collisions are, for reasonable size passwords, are going to be very rare. But why would a system check the hash? So what it's clearly doing is looking up a user's record with the password as the index. And it finds an existing one and says, oh, sorry, we can't have - two users cannot use the same password because then our index will return two records, and we can't have that happen.

So, I mean, it just - it's mind-blowing that somebody could say, oh, sorry, that password is in use. Well, now, of course, that also begs the question, ooh, I wonder who's using it. Because of course it's an information leak, as well. So, I mean, bad architecture, fundamentally flawed account management, it's just horrendous.

Leo: And we should add, if you ever see this, you should never see it for two - not only that, but because it means you have a crap password.

Steve: Very good point.

Leo: Right?

Steve: If your password is colliding with somebody else...

Leo: You've done a bad job, my friend. You've got to stop using Monkey123.

Steve: That's a perfect point, Leo.

Leo: You should never, ever see that image.

Steve: So anyway, thank you, everybody, for sending that to me.

Leo: That's pretty funny.

Steve: I just - that was wonderful.

Leo: It's pretty hysterical.

Steve: And we've got super astute listeners who've, you know, been following along.

Leo: They got it.

Steve: Yeah, they got it.

Leo: Yup.

Steve: So the first story was covered by Forbes. And I'm looking now at the date to make sure it's recent.

Leo: Well, you know, and by the way, this is a lesson for anybody who snarkily says, well, heh, heh. So right after I said that to you, I pulled up your show notes because I was going to show the image, and it didn't match what you were saying. And I thought, well, this is strange. Then I realized the show notes I had were from 2011. So somehow 2011 is back, baby. Now I have the real show notes up.

Steve: So anyway, Forbes' title is rather brutal: "Feds Walk Into a Building, Demand Everyone's Fingerprints to Open Phones."

Leo: Wow. No.

Steve: And so I'm just going to read the top of the story, and two wellknown legal experts were consulted for their opinion. So Forbes wrote: "In what's believed to be an unprecedented attempt to bypass the security of Apple iPhones" - although actually the warrant specified four different brands, all known to have fingerprint readers. So the law enforcement officials didn't even known which phones people might have, so they just blanketed it. I mean, and that's the big concern. Not only is there a concern with forced use of biometrics, which the argument has been is not constitutionally protected, but in this case it's also just the breadth of this warrant.

So "the security of Apple iPhones or any smartphone that uses fingerprints to unlock, California's top cops asked to enter a residence and force anyone inside to use their biometric information to open their mobile devices. Forbes found a court ruling dated May 9, 2016, in which the Department of Justice sought to search a Lancaster, California property. But there was a more remarkable aspect of the search, as pointed out in the memorandum: 'authorization to depress the fingerprints and thumbprints of every person who is located at the subject premises during the execution of the search and who is reasonably believed by law enforcement to be the user of a fingerprint sensor-enabled device that is located at the subject premises and falls within the scope of the warrant.' The warrant was not available to the public, nor were other documents related to the case.

"According to the memorandum signed off by U.S. Attorney for the Central District of California, Eileen Decker, the government asked for even more than just fingerprints: 'While the

government does not know ahead of time the identity of every digital device or fingerprint (or indeed, every other piece of evidence) that it will find in the search, it has demonstrated probable cause that evidence may exist at the search location and needs the ability to gain access to those devices and maintain that access to search them. For that reason, the warrant authorizes the seizure of "passwords, encryption keys, and other access devices that may be necessary to access the device."'

"Legal experts were shocked at the government's request. Marina Medvin of Medvin Law said: 'They want the ability to get a warrant on the assumption that they will learn more after they have a warrant. Essentially, they are seeking to have the ability to convince people to comply by providing their fingerprints to law enforcement under the color of law because of the fact that they already have a warrant. They want to leverage this warrant to induce compliance by people they decide are suspects later on. This would be an unbelievably audacious abuse of power, if it were permitted.'"

And then Jennifer Lynch, who's the senior staff attorney at the Electronic Frontier Foundation, added: "It's not enough for a government to just say we have a warrant to search this house, and therefore this person should unlock their phone. The government needs to say specifically what information they expect to find on the phone, how that relates to criminal activity, and I would argue they need to set up a way to access only the information that is relevant to the investigation. The warrant has to be particular in how it describes the place to be searched and the thing to be seized, and limited in scope. That's why, if a government suspects criminal activity to be happening on a property, and there are, say, 50 apartments on that property, they have to specify which apartment, and why, and what they expect to find there." And then she concluded, saying: "We've never seen anything like this."

Now, nobody really, because this is not public, nobody understands the details. But from our technical perspective, if we were to sort of reverse-engineer what may explain this - because I was thinking about, like, okay, first of all, the concern is this is now what law enforcement - this is apparently law enforcement's maybe another test. But their reaction to the famous iPhone fingerprint hack problem, where they're just saying, okay, fine, we're just going to, since the courts have said a password constitutes - coercing a password constitutes self-incrimination, which is protected by the Constitution, we can't do that. But the courts have also previously said biometric information is not protected. So law enforcement is just saying, okay, fine. We want specifically the warrant to give us the right to force anyone in this location to unlock their phones. So again, the generality of this makes me think it's tied to an IP address. That is...

Leo: Ah.

Steve: Right?

Leo: Interesting, yeah. That makes sense. They know it's this building.

Steve: Or this residence. It was actually a house. And so they probably found some traffic which they believed represented, whatever it was, criminal activity. They got the address from the cable provider or whoever provides Internet service. So now they know that it is a device, and they just presume probably a cell phone. If not, maybe a PC, but those don't tend to be locked the way mobile devices are. And so they wrote the warrant to give them permission when they get there to force everyone to let them look inside their phones, presuming that one of them will have been the endpoint of that electronic communication.

Leo: Yeah. Wow.

Steve: Which in my mind doesn't excuse it, but it sort of explains, like, probably...

Leo: What they're up to, yeah.

Steve: Yeah, what was going on.

Leo: So then that begs the question, because all of us now I hope have fingerprint-protected phones, what you should do. And it seems to me the easiest thing would be, when the feds pound on the door, besides flushing all the drugs down the toilet, you want to turn your phone off because at least all the phones I've used, iPhone and Google phones, once you turn it off, the fingerprint won't re-unlock it. You'll have to then enter your passcode.

Steve: Right.

Leo: Which they can't reasonably demand. I mean, they can demand, but not reasonably.

Steve: Right. That, or call your attorney and impose a 48-hour delay.

Leo: Oh, there you go. And that'll do it, too.

Steve: Because that's the other thing that will cause that to expire. So, yeah. Again, we're in some interesting uncharted territory. I will say that Apple seems to have solved with iOS 10 the problem of the fingerprint reader working too quickly. I hate this double-staggered...

Leo: I have a fix for it, if you want.

Steve: Oh, do you.

Leo: Yeah.

Steve: I looked for one, couldn't find one.

Leo: I do, too. I hate it, too.

Steve: I just - I hate it.

Leo: So that's something new in iOS 10. And the reason - I figured out why they do that. Because a long press of the fingerprint opens Apple Pay. So they want you to be able to use Apple Pay. So what they've asked is that you have to unlock it and then press it again. But if you go into the accessibility settings, you go to Settings > General Accessibility, and then there's a Home button setting. You can restore the original pre-iOS 10 behavior of it both unlocking and...

Steve: It goes right in.

Leo: Goes right in.

Steve: Oh, good.

Leo: So it's in the Home button settings of Accessibility. Not the general Home button settings. You have to go into Accessibility. But, yeah. As is often the case, they hide stuff. They hide the stuff you really want under Accessibility. But if you...

Steve: Yeah, well, I always turn up the visibility to - because I don't like that whole color bleed-through thing.

Leo: Right.

Steve: I get rid of the animation...

Leo: Exactly, yeah.

Steve: ...because I don't need things zooming up and down and all that.

Leo: Oh, but you're not going to get, then, the cool fireworks in the iMessages. All right. So here it is: accessibility home button. You have click speed, but you also have rest finger to open. And that, if you turn that on, it restores the previous...

Steve: Oh, good.

Leo: You don't need to do the double-step tango.

Steve: Ah. It just - and it's funny, too, because, I mean, I just - I liked being able to press a button, and I'm in.

Leo: Yeah, that's the point.

Steve: Instead of now you kind of have to wait for it to say it, or you press it again, or it's just so aggravating.

Leo: It is, yeah.

Steve: So, yeah, I'm glad they have a workaround.

Leo: Yeah.

Steve: Okay. Now I want to take our listeners through what Quarkslab found when they did what could only be considered a really thorough audit of VeraCrypt. Okay. I will step on the conclusion here and just say that you want 1.17. Or is it 9? Let me look. Nine, 1.19. What these guys audited was 1.18. So most of the problems, but not all of them, are fixed. And I think one of the reasons that so many people sent links to me, breathlessly, is that, once again, the headlines in the popular press show big red warning flashing, you know, "Emergency upgrade right now." And nothing here was a real showstopper. We've been living with it as it has been for a while.

But this is a perfect case for us on this podcast to take a look in detail at the nature of these sorts of problems. So as I mentioned before, this came about as a result of a small group who call themselves OSTIF, the Open Source Technology Improvement Fund. And they sort of have three missions. One is to offer bug bounties to help improve code, direct code improvement through grants, and professional audits. And so, due to the history of TrueCrypt, and the fact that VeraCrypt took it under their wing and then have moved it forward, they felt it was time to give this thing some real scrutiny.

So 10 weeks ago, from around today, on August 1st, the OSTIF announced: "OSTIF is proud to announce that we have come to an agreement to fully fund an audit of VeraCrypt. Using funds that were donated by DuckDuckGo and VikingVPN, we plan to hire Quarkslab to go over the code and search for vulnerabilities and backdoors. VeraCrypt is a crucial piece of open source software that can encrypt any storage medium with powerful and highly tamper-resistant encryption that greatly enhances the personal security of anyone who uses it. An audit of the code brings fresh professional perspectives and a deep analysis of the code to search for vulnerabilities." And it goes on like that.

So basically, 10 weeks ago, this was contracted. And after they began, 32 man-days later, we have results. So, okay, first of all, what was found? I would argue no huge showstoppers. But the devil's in the details, and we'll go over that. But they did find a handful of both old and new things that needed to be cleaned up. Most of these were weaknesses that could sort of theoretically be used in highly targeted attacks to compromise the target's privacy and security. But mostly they are things that fell out of the technical scope of coverage. That is, even TrueCrypt noted some problems and said, "But that's somebody else's problem." That is, this is not the kind of thing it's feasible for us to protect against. But there are some things that were feasible that we're still not protected from. So there are both troubles inherited from TrueCrypt in VeraCrypt, and then there were new troubles introduced by new VeraCrypt features.

So, and it's funny, too, because, I mean, any of our longtime listeners, a lot of this will be very familiar because, step by step over the years, as problems were found in TrueCrypt, we talked about it on the podcast and covered it in some detail. So as I was reading through the audit, I'm going, oh, yeah, I remember that one. Oh, yeah, I remember that one. So one problem was, from TrueCrypt, their password-based key derivation function was not acceleration resistant. That is, they didn't use a so-called "memory-hard" function like scrypt. It was just an iterated hash. And we know from all of the work that's gone into minting bitcoin and other crypto currencies that you can apply especially large-scale integration to hugely accelerate those operations.

So VeraCrypt was using, or VeraCrypt inherited from TrueCrypt, iterations of either 1,000 or 2,000, which depended upon which hash algorithm was used and its usage, that is, whether you were mounting a volume within the system or essentially accepting a password to boot the system from an encrypted system partition. So the original iteration counts were 1,000 and 2,000. They are now - get this - 200,000 and 655,000. So way stronger.

There's also an additional feature that VeraCrypt added, but introduced a bug that we'll cover later known as a PIM, in this case a Personal Iterations Multiplier. And that gives the user some additional control so that, if they don't mind waiting longer at boot or mount time, they can specify a so-called Personal Iterations Multiplier which, as it sounds, allows you to dramatically multiply that even further. Problem is, turns out there was an overflow in the math for the dramatic multiplication so that it ended up reducing the rounds count, rather than increasing it, when you thought you were getting more security. Whoops. Anyway, that was easy to fix and was fixed.

There was also a problem from TrueCrypt that sensitive information might be paged out from the kernel stacks. So you've got a kernel driver where the encryption/decryption is going on. It has to have an operating symmetric key present, which means that key is persistently vulnerable to exposure. If, in a low memory situation, a page of the kernel gets swapped out to non-encrypted storage, your key might be there. So that's a concern. However, that was the case in TrueCrypt. VeraCrypt doesn't change it. They simply state that this will be a vulnerability unless the system partition is encrypted and the paging file is therefore written to an encrypted partition. So, and that's, I guess, that's some protection. It means that the key won't be readable. But you really don't want the key to go out into the swap file because that's potentially accessible, too.

In the details of this they talked about the fact that, unfortunately, both of these programs, the original TrueCrypt and VeraCrypt, which inherits most of itself from TrueCrypt, has sensitive data scattered around. So it is very difficult to collect it all in one place. As an aside, that's one of the things that I did very carefully, like from day one of the SQRL client for Windows, is I created a protected memory region, and absolutely everything that matters is within one solid contiguous block. And I also arranged to lock it into memory. And the program is so small, there's no danger of it being swapped out because programs all get a chunk of memory that isn't in danger of being swapped out. It's only if they get really huge that then they're at risk of having some other pieces removed.

So anyway, so this is the problem with trying to do sort of add-on security to an operating system after the fact. You are fighting some of the architecture of the system. And it tends to be an OS by OS thing, too, because of course this is all multiplatform. And the auditors did acknowledge that this is a difficult problem to solve on a multiplatform system, or a multiplatform solution, because it requires deep knowledge of, not just one OS, down at the kernel level, but Windows and Mac and Linux. Wherever it's going to run, it's a significant implementation challenge.

Another problem that they did fix, but that TrueCrypt inherited from - I'm sorry, that VeraCrypt inherited from TrueCrypt, was the various uses of compression and decompression. TrueCrypt used and sort of forked some early Zip and Unzip libraries which had known problems. And VeraCrypt didn't change it until this audit brought that to the attention and said, you know, even though exploitation is unlikely, why keep using these known problematical libraries? So the idea would be that, if it were ever possible for an attacker to manipulate what was being decompressed, then that could be used to create a buffer overrun and potentially execute some code under the attacker's control.

So the reason this wasn't considered - oh, and I should mention this was brought up in '08, so eight years ago. And the TrueCrypt guys knew this and said, well, you know, again, this is sort of out of scope. So an example of an attack would be the recovery disk is compressed, as is the boot code, because it has to be so small in order to get the system going. But it's unlikely the boot code could be manipulated. The recovery disk, though, could theoretically be manipulated by an attacker so that, when a user used their recovery disk, it would decompress. And if the recovery disk had been modified, the decompressor could have an overflow induced in it that, for example, might write the user's password to some external location or some fixed location where the attacker could then access it.

But the counterargument is, okay, on the other hand, if a bad guy has that kind of access, the ability to alter the recovery disk, then they could do a lot more than something as awkward as manipulating the decompression problem. So essentially the auditors were saying, "This is bad, and we're trying to figure out why." But it's the kind of thing where - and we see this all the time in security - vulnerabilities that are not obviously exploitable initially, even when they're identified, oftentimes in the future, due to changing circumstances or processor power or whatever, suddenly they then become exploitable. So the compression/decompression has been fixed.

We also talked about, and VeraCrypt had fixed and these auditors verified - and you'll remember this one, Leo, where - TrueCrypt is written in C. And the authors were using the memset function, which basically allows - memset is a C function or a library function which allows you to say "Set the following block of memory to this value." And typically it's zero. And so you want to zero out or null a block of sensitive data like the password, or like the master key or something. I mean, and so secure systems are often having to have transient, highly sensitive data, and they're often allocating it dynamically. So you say, "give me a buffer." You ask the operating system for a block of memory. And it says, okay, here's a pointer to the block of the size you requested. You're then able to use that until you free it, that is, you essentially tell the operating system, "I'm done using that block of memory, thank you very much."

So what the programmers of TrueCrypt did was they were careful, because they don't know what the operating system is going to do with the memory that they give back to it, they were careful to zero it, to write zeroes across the buffer, then return it to the operating system so that memory that might have had something sensitive didn't just go floating around and may be available to somebody else. The C compiler, though, as one of its optimization strategies, is looking for things that don't do anything. And some clever programmer somewhere said, oh, look. This memory is being zeroed and then immediately returned to the operating system. Well, so there's no purpose to zeroing it. That must have just been a mistake. And so the C compiler optimized out.

Leo: They didn't want to do that.

Steve: Yeah, that serves no purpose. You've zeroed the memory and given it away.

Leo: Isn't that a funny error. Wow. Too helpful.

Steve: Yeah. So there is a secure zero memory function which doesn't have this problem. And so one of the early things that VeraCrypt did was to switch over wherever memset was being used to this secure zero memory function which doesn't risk being optimized to nothing by the compiler.

And there were too many subtle TrueCrypt kernel driver problems fixed to mention that were fixed by VeraCrypt since it inherited TrueCrypt. And I won't go through them all. I mean, there were just a collection of little subtle things. But the takeaway is these guys, the auditors, really did a nice job. There's no way you could come away from this thinking, wow, this hasn't been scrutinized deeply.

There was one that was sort of interesting where they found that the kernel driver was not checking access permission on behalf of the user requesting to see whether a file existed. Now, the kernel, of course, can do anything. It's the god of the local environment. But it has to respect access permissions. And like I said, it has to enforce access rights permissions. In this case, that was deliberately left out by the TrueCrypt developers because they wanted the boot time, or they wanted an unprivileged user running TrueCrypt to be able to determine whether there was boot code on the system that would be privileged. So the kernel driver was specifically allowed not to check access permissions.

What was again found by a close examination, and you have to take an adversarial position, which is what these guys did, is that you could use that, you could leverage that as an information leakage, allowing anybody to probe the system for the existence of files for which they have no access. The system should say, "You have no access." Instead it says, "Yeah, that exists"; or, "No, that doesn't exist." And so this could be exploited for an information leakage problem. And this is an example of something VeraCrypt had not addressed. And so these auditors brought that back to their attention and said, yeah, you know, you need to think about this. Also, the AES cipher, which is sort of the de facto default cipher, has cache timing vulnerabilities, if the chip it's on does not support the AES instruction set.

Now, the mitigating factor here is that Intel has supported the AES instruction now for quite a while. And maybe really low-end chips or old chips might not have it. But mainstream strong desktop chips have had it for years. So if the hardware instruction-assisted code is used, then there's no problem. But the system runs on processors without the AES acceleration instruction. And in those cases this software is not cache timing proof, which is something that these guys again found.

One thing that we talked about in the past was that the TrueCrypt developers misused the cyclic redundancy check. CRC-32 was never designed to function as a cryptographic integrity mechanism, and they kind of crossed the line. They use it that way. What it was designed for and what it's really good for, is efficient error detection. But that's different than integrity mechanism, and the difference is it's intended to prevent accidents, not thwart attacks. And so as a consequence of the fact that TrueCrypt does - and VeraCrypt has continued for the sake of backward compatibility because unfortunately it's built into the volume headers and various signatures, and it's very small and very fast, which are both reasons that they used it.

The problem is it is subject to manipulation. So, for example, the CRC-32 is used to mix the keyfiles in. Remember that TrueCrypt introduced this notion that not only could you make the master key that you decrypt dependent upon a user-provided password, but you could also use a keyfile, the idea being that it's just going to be some file unknown to an attacker with presumably a lot of entropy. And why not just pour that in? Well, that would be nice if they had run it through a good hash. Unfortunately, they ran it through CRC-32, which is explicitly not a good hash. You know, it's a horrible hash. It's not a hash. And so from a valid set of keyfiles it's possible to create another distinct one file which will also mount a volume. Which is something you really don't want.

One of the properties of a hash is that it's not easy to compute, well, it's really, really,

really hard to compute an outcome; to figure out what to put in in order to get something out. And so what that first example is is that the CRC-32 doesn't provide that same guarantee. It's also possible to create null keyfiles, that is, a big file that does nothing to modify the so-called "keyfile pool." And it's even possible, given a set of keyfiles, to remove the security brought by that set of keyfiles, essentially to add another file that completely nulls the effect of all previous ones. Again, something you could never do if you were passing all this through a real hash function.

And finally, from a known passphrase, it's possible to create a keyfile which removes the security provided by the passphrase. And again, this problem, too, has been known since 2008, for eight years. But the TrueCrypt authors just decided they didn't feel this was a [audio dropout] because it required prior knowledge of a secret or the prior manipulation of a machine. And so here's, again, this is like, well, okay, yeah. But, boy, if you just used a hash, then you wouldn't have these problems. And this is the kind of thing that, as we've seen before, that tends to come back and bite people downstream when some clever person figures out how to use that special case in a way that just happens to fit their needs.

Also, volume headers are "authenticated" with CRC-32, which is weak authentication. Mostly it's just a means for the TrueCrypt driver to validate that what it's being given is a TrueCrypt volume. And so it's not clear why using CRC-32 opens a vulnerability, but it just seems sloppy. Also, James Forshaw, who's with Google's Project Zero, he discovered, and we reported back when he did discover this, a pair of problems in the kernel driver. One was a drive letter symbolic link creation elevation of privilege, and a lesser concern problem. Both have been fixed, and these auditors verified that.

Here's an interesting one that I just really appreciated due to its subtlety. And again, it just shows you that these guys thought about the code. The auditors really thought about this. When booting from a BIOS, and that could be either UEFI or traditional BIOS, it turns out the user's password length can be obtained. And we know that's not good because that makes brute-forcing substantially easier. You don't have to bother, if you know the password length, you don't have to bother with trying any shorter passwords or any longer passwords. So it doesn't, like doesn't make it instant, but it dramatically simplifies the problem.

The BIOS saves keyboard keystrokes in a 32-byte, 16-character ring buffer. Essentially, every key you press puts two bytes in the buffer, the scan code of the keyboard and its ASCII equivalent. And it's a ring buffer, meaning that - and this is a common programming technique. When you get to the 30, well, to the last word, the last two bytes, then when you go to the next one, essentially it wraps around to the beginning. It goes back to the start of the buffer. And so, and this is sort of - this is commonly done in programming. It allows you to sort of have the equivalent of an infinite buffer, as long as somebody is reading from the end fast enough to keep the front from wrapping around and overwriting the oldest data.

So these buffers are managed, ring buffers are managed with a typically called a "head" and a "tail" pointer. The head is the pointer to the last entered character. And so when a new character gets entered, that pointer is advanced or wraps around to the beginning, and then the characters are written there. And then the tail pointer is the character next to be read by somebody requesting the next character from the buffer. So if the head and tail pointers are equal, that is, essentially that means that the reader has read everything that's available in the buffer. And so when another character gets pushed, the head pointer moves forward. And then if the software pulls because it's looking for input, it notices that the head and tail are different, meaning that the head has moved further forward. So then the tail is advanced as characters are retrieved.

Well, it turns out that the authors of this code - and it wasn't clear to me whether this was VeraCrypt in the beginning or the original TrueCrypt. But somebody recognized there was a problem, and they explicitly cleared the buffer. We know where that buffer is in the BIOS. It is in a universally fixed location in RAM. And so TrueCrypt zeroes it. But they forgot to zero the head and tail pointers. Those are two words immediately in front of the buffer, that is, the lower addresses. And those are not being cleared. Never have been.

Which allows an attacker to determine, with mod 16, essentially, the length of a user's password. I don't expect many people are entering passwords longer than 16 characters. Maybe they are. In which case, that would still give an attacker a clue. If it looked like it was four characters, it might have been 20 characters. So that would still allow them to hugely decrease their search space in brute forcing. And so these auditors picked up on this problem and said, eh, you know, the head and tail pointers is another little information leak that should be taken care of.

Also, the VeraCrypt guys added command line options to the TrueCrypt invocation. Unfortunately, you can give it the volume password. And VeraCrypt added support for smartcards. Smartcards require a PIN in some cases. And you can also put that on the command line. These auditors thought that was a really bad idea because there are systems, for example, command line stacks that would keep that separately in the system. So there you've got, potentially, your TrueCrypt or your VeraCrypt volume password sitting in the past command line history stack and available to anyone who knows to look for it. So they've strongly objected to the provision of secure information like that on the command line.

And one of the other problems is that VeraCrypt added four new non-Western cryptographic algorithms, three ciphers and a hash. And one of the ciphers and hash were Russian design. The cipher is old, dating back from the 1970s, called GOST89. And while it has a very nice 256-bit key, its block size is only 64 bits. And just recently on the podcast we've been talking about the inherent problem with 64-bit block ciphers. They're just - they don't allow enough permutations of the bits in order to be secure. And because the VeraCrypt guys apparently so much wanted to use this Russian-designed cipher, just I guess for the sake of internationalism, I mean, there's nothing wrong with AES that everyone can use, which is 128-bit block cipher. They created sort of a CBC, a cipher block chaining variant, to concatenate two 64-bit cipher iterations in order to sort of kludge 128-bit equivalent.

Unfortunately, it turns out that many of the security guarantees of cipher block chaining fail when you always have a null initialization vector which they had to have because there was no provision for having unique initialization vectors in this application. So the good news is the auditors understood all of that, and GOST89 has been removed from VeraCrypt. There just wasn't a safe way to essentially backport an old cipher with a small block size. It was just a bad idea. So again, another reason why having lots of eyes on these problems is a good thing.

And there were another bunch of problems, I won't go into detail, that the auditors considered as informational, you know, sort of things that should be cleaned up, but weren't crucial. However, it is worth noting that the UEFI bootloader - I talked about the BIOS key buffer problem. It turns out that that may even be worse with UEFI driver. UEFI drivers maintain their own keystroke buffers; but, unlike the BIOS, the buffers' internal address and location are not known. There's an API that you use, and there is no visibility past that API.

So the console in API does offer a reset method. But there's no guarantee of what that does in any specific implementation. So it may just, for example, it may be to empty the buffer. But emptying a ring buffer means setting the tail pointer to equal the head pointer. So that may be what reset does is just align the two pointers, rather than zeroing the memory. It's really feasible to imagine that that's all it does. And so it turns out that, in examining the Intel Developer Kit sample code for UEFI, it turns out that the Intel sample UEFI ConsoleIn driver calls Int16, which is the old BIOS ring buffer. And so it's added another layer of abstraction, but the fundamental problem is still there. And maybe it's got its own internal buffer. We don't know.

So this is interesting because here's a collision of low-level operation at boot time before the OS is running, where we're relying on very old, I mean, like, you know, from day one of the IBM PC, that's where the BIOS first appeared, and Int16 appeared, and we had the clanky IBM keyboard, and it had the 16-character buffer, and it's still there today. The problem is back then we weren't trying to boot TrueCrypt volumes. And so there was, like, no expectation that what was being typed in had to be super secure. And, if it had to be, IBM published the source code for the BIOS, and so everybody knew where the buffer was. It never moved. And so you could zap it if you needed to.

But now we've, like, moved several generations beyond that, and we're imposing requirements for security that the underlying architecture was never designed to explicitly support. And so it's a fundamental architectural problem, rather than a mistake that anyone made. And I guess the only thing you could do would be to service the hardware yourself. But that's a problem that SpinRite has, for example. The reason it doesn't currently run on a Mac natively is the Mac doesn't have a BIOS. It uses a USB keyboard and emulates that. SpinRite in one area of its code assumes it's on a PC, and it does its own hardware keyboard reading because it needs some of the facility that that provides. And so that's, for example, something I had to back off from when I was beginning to work on 6.1, in order to allow it to run over on a Mac without any problems. So not easy solutions.

So the audit concludes: "This audit, funded by OSTIF, required 32 man-days of study. It shows that this follow-up of TrueCrypt is very much alive and evolves with new functionalities like the support of UEFI. The results show that evaluations at regular intervals of such difficult security projects are not an option. When well received by the project's developers, they provide useful feedback to help the project mature. The openness of the evaluation results help build confidence in the product for the final users." And I think that's 100% correct.

So essentially these guys verified that many old problems had been found and fixed. And they looked carefully at the new things that had been added and found some problems that needed to be and have been fixed: GOST89 was removed; the compression and decompression algorithms were replaced with secure zip libraries; that password length leakage and the buffers are being cleared to whatever degree is possible; and other bootloader vulnerabilities were resolved. So VeraCrypt is now at v1.19 and is even more Vera than it was before.

Leo: Sounds like it might even be better than TrueCrypt was. I mean, a lot of these were TrueCrypt flaws.

Steve: Yeah, I agree. Now, again, it was never our sense that there was a critical gotcha in TrueCrypt. But we did say, oh, was it about a year ago, I think it was when the Project Zero findings came out, it's like, okay. Because it was a problem in the kernel driver. There was an elevation of privilege that any process in the system could leverage because there is no way to restrict driver access to a specific process. So it just, it was like, okay. And we said on the podcast, eh, it's probably time to migrate. Certainly if you're setting up a new system, use VeraCrypt. Don't stay with TrueCrypt. It's a superset. And of course moving forward it's becoming more important to be able to have support for UEFI for modern hardware platforms. And VeraCrypt continues to do that, and TrueCrypt just won't do it at all anymore.

Leo: Yeah. I'm also pleased to see that it was audited so quickly in its life compared to TrueCrypt, which we used for years without really any audit.

Steve: Yes.

Leo: I think that's great. I mean, this is something a lot of people are going to rely on.

Steve: Yeah. And if nothing else, as I said, this just overwhelmed my Twitter feed because so many of our listeners said, wow, okay, what does this mean? And so it's obviously of huge interest to people. And I'm probably fine with BitLocker because I don't have...

Leo: Microsoft's solution.

Steve: Exactly, Microsoft's built-in solution, where it's available. It's not available on all platforms, not the lower-end platforms.

Leo: Windows Pro, yeah.

Steve: Yeah. But I fully understand that there are people who like the idea of a third-party solution, and one that is open and has received this kind of scrutiny. This is what this kind of software needs.

Okay. So what we talked about, I don't remember which of the three weeks it was, Leo, when you were away, and Father Robert and I were doing the podcast, was it came to light that both Firefox and Chrome are frantic about taking snapshots of the current browser tab space constantly, and to the tune of gigabytes redundantly written to the system drive per day. And in fact, if you take the write endurance of an SSD, that is, the amount that you write to it per day over its life, leaving Firefox or Chrome open, even on an idle system, can be consuming half of the total write life of the SSD.

So one of the things that we talked about the first time this came up was that Firefox does offer a tweakable setting under about:config. And we've talked about that little amazing zone of configuration many times. But you put about:config in the URL, and then you need to use a search term because there are so many options. But browser.sessionstore.interval is the item. And it's in number of milliseconds per refresh. It defaults to 15 seconds. And what it must be doing, and this is what's sort of disturbing, is that apparently they just didn't try to optimize this at all. If they had done any optimization, they would, for example, ask the question first: Has anything changed in the last 15 seconds? Apparently they don't. And so they just take a snapshot of the entire context of the browser and write it down into your session profile. And not only Firefox, but Chrome does the same thing. In fact, I think Chrome was even a little - it was writing a little more.

Now, this was of huge interest to many people. I have an SSD-based laptop, and we know that the Windows 7 machine that I recently built has an SSD as its system drive, with spinning drives in a RAID as backup. And so I am absolutely going to do what I think is a better solution, and that is, at least for Firefox - and I haven't explored this for Chrome. But for Firefox, under application data > Mozilla > Firefox directory, there's a profiles.ini. It's a short little INI file, about seven lines, and one of the lines is IsRelative. You need to set that from a one to a zero, and then there's a path where you can specify where these profiles are written. Just change that to a spinning drive, that is, you know, not your system C, typically, SSD, but I normally use Z or something, like something at the other end of the alphabet as my archival storage. And so I will absolutely create another non-write-fatigable place for my browsers to write their profiles. And I will also hope that they get smarter about this because it's just crazy how much data that they're writing constantly.

Leo: I'm surprised, given that almost everybody at Google, presumably at Mozilla, too, is probably using computers with SSDs...

Steve: Yeah.

Leo: ...that they haven't fixed this. I'm sure they will.

Steve: Yeah. Again, it sort of feels like it just didn't come up. The architecture's been there since before SSDs, and until somebody pointed it out - what happened was they used Mark Russinovich's Process Explorer. And so anyone could do this, and we talked about this a couple weeks ago. You just run Process Explorer, and under the View item there are columns. You're able to turn on more columns in the display. And there's Disk I/O. You go to that tab, and then it's Bytes Read and Bytes Written. You turn both of those on, click Okay, and that adds those columns to the display. And if you just sit there watching it, you just see Firefox or Chrome [sound effect], just going crazy.

And so, you know, my immediate reaction, although I'm on spinning drives on this old system of mine, on a RAID, was to just slow it down by a factor of 10. I added a zero. It was 15,000, meaning 15,000 milliseconds. I made it 150,000 so instead of every 15 seconds it was every 150 seconds. But a better solution is just move it to somewhere else. And if you don't have a somewhere else, like for example on this nice laptop that I've got that is SSD, I think I may go back to the days of a RAM disk because it's got a ton of RAM. So I'll just create a D: RAM drive and have the profiles live there.

Leo: You'd think that the controller and the caching software would handle all this and not pass it on directly to the hard drive.

Steve: Well, but anything you - so it could, if it was reading it. But the problem is it has a write-through cache.

Leo: Has to write; right.

Steve: Yeah. Because the whole point is, if the system, if the browser crashes or hangs, or the system crashes, you want to be able to restore your session. So this is all just session restore. But, boy, what a cost. Again, it has to be because they just didn't give it any attention. And I'm hoping that, now that this has come to light, that somebody will spin off a little side project and get busy and fix this.

So the setting is the Recode Code Conference, 2016. Elon Musk is onstage with interviewers Walt Mossberg and Kara Swisher, taking questions from the audience. Josh Topolsky, who is a co-founder of The Verge in the audience, asks Elon whether he thought we might be living in a simulation. We immediately learn, to our surprise, that Elon has ruminated about and hashed through this topic so thoroughly already that he and his brother have formally banned its further discussion anytime they are in hot tubs. Its discussion is formally banned from hot tub debate.

And he was dead serious about this. So in answering the questioner, Elon put a number to it and even, like, worked through his logic, concluding that he believed we are indeed living within a simulation. And he says, in fact, he thinks there's only probably about a one in a billion chance that we are not simulated and are living in what - that is, a one in a billion chance that we are living in what he refers to as a "base reality."

So it turns out that this is - and maybe Josh knew this already because Elon was on record, at least for a couple years. Back in 2014 at Vanity Fair's 2014 New Establishment Summit, Elon made the case that our lives are not at all what we think they are. He concluded, again, even back then, there's a one in a billion chance that this is reality. And so he said, for example, quoting from this Recode Conference recently, he said: "The strongest argument for us being in a simulation is the following: 40 years ago we had Pong, two rectangles and a dot. Now, 40 years later, we have photorealistic 3D with millions playing simultaneously. If you assume any rate of improvement at all, then the games will become indistinguishable from reality. It would seem to follow that the odds that we're in base reality is one in millions."

Anyway, so I thought, okay, so that's interesting. Now, a lot of the press jumped on "The Matrix." And what I realized was that, okay, that's sort of the pop version of what this sort of sounds like, but it's not what Elon means. So the difference is "The Matrix" we can think of - everyone knows the movie, of course. We can think of it as sort of an MMO; right? Just it is a massively multiplayer environment, where you are sort of a reflection of yourself in this other environment. What Elon is referring to is different than that. He's referring to more something like the Game of Life, of course Conway's famous game, where you have rules by which these little cells are born or live and die. And those rules create this surprisingly rich result.

And so what Elon is meaning is that everything, everything, what we regard as reality, could be running in some sort of a reality support container, meaning that we, life, are an emergent property of that. So it's not that we exist as ourselves outside of this, as was in the case with "The Matrix," but in fact that reality as we know it is something that we perceive from inside, but that it exists in some sort of a container. And Leo, I heard you talking about it, I guess over the weekend. And there was some conversation about continuity, like the continuousness of space. And what is a little creepy is that it turns out that most recent theories of space time is that it is quantized. That is, photons do not travel continuously at the lowest level. They are jumping from one, essentially, pixel, for lack of a better word, to the next. So that's kind of creepy. Anyway, I just - I thought that was a kick and wanted to stick it into the podcast.

So, and I did, when I was trying to think of an example of this, Star Trek followers will remember in the Next Generation series the creation of Professor Moriarty on the holodeck, where Data was role-playing, essentially, as Holmes, Sherlock Holmes, and asked the computer to create a rival worthy of Holmes in the form of Professor Moriarty, and a conscious entity was created. I think it was entitled "Ship in a Bottle," I think was the title of the episode. And everybody felt bad about having created this consciousness who, like, apparently did exist. But they ended up giving him his own reality which would continue running, so at least he'd be happy in his little world that was all a program. Cool stuff.

Leo: Well, there you go.

Steve: So a little quickie on the lithium-ion battery issue. And Leo, I know you've been interested in this. I've got two pictures. I found a guy, thanks to a follower who pointed me at a link, and I followed it down. There is a company, Qnovo, Q-N-O-V-E, dotcom. The guy there has a blog, and this person is in the middle of this battery technology. For example, the most recent blog posting is titled "The Perils of 4.4 Volts." Before that, "The Real Science Behind Battery Safety." Before that, "It's Easy, but Not Right, to Pick on Samsung." Before that, "Making Sense of 100 kWh." And, for example there, I picked up on this because it mentions Tesla. He says: "Tesla Motors announced today upgraded versions of the Model S and X boasting 100 kWh battery packs, up from 90 kWh used in their earlier top-of-the-line electric vehicles. One hundred kilowatt-hours sounds like a lot, and it is; but I bet that many readers won't have an intuitive sense of this amount of energy. This is what this post is for."

Anyway, this guy, to sum up - and I've got links in the show notes for anyone who wants to dig in more. Or you just go to Qnovo.com, and you'll find at the top in the menu a link to his blog. And it's really good stuff. He explains that the problem that these batteries are experiencing is the formation of lithium metal crystals which, due to a combination of physical manufacturing problems and pushing the batteries as hard as we do these days, can force the lithium crystal to grow through the insulator and create a lithium bridge between anode and cathode. Then you have a short circuit, and the battery dumps its energy into itself, essentially, and explodes.

And so one of the things - and this is, I mean, this is well understood. But the solution for this is to guarantee that the cathode is physically smaller than the anode. That is, that the anode, the edges of the anode extend beyond the cathode. And that simple physical property prevents the formation of these crystals, or mitigates the problem. But that extension, while it creates a safety margin, it does not store energy. So in the pressure to get the energy storage as high as possible, the manufacturing engineers have cut the margins. And you can imagine the pressure on battery makers for their batteries to, I mean, that is the pressure is we want them to charge fast, and that's dangerous; and we want to charge high, that is, right up to their maximum voltage because, if we aren't charging it as high, we're not putting as much energy in. And so the combination of speed and extent of charging, then weakened by this loss of margin in the oversizing of the anode, ends up creating a catastrophe.

And so his point is Samsung was only the first to get bitten. And of course remember years ago, he says 15 years ago laptops were having this problem because there were problems with laptops back in the day. And as we talked about a couple weeks ago, last Christmas the hoverboards were catching fire and exploding. So it's a problem, a combination of manufacturing and that we're pushing the margins. We have pushed the safety margins out in this quest for storing ever more amount of energy in a smaller space. And wanting quick charging, which is challenging, as well. So I just wanted to kind of end that topic, follow up on it, because I found this terrific site.

I did want to briefly mention that I spent some time this weekend and made a major update to the Healthy Sleep Formula page with a complete dosage determination protocol. Many people have been reporting success, but they just didn't have any guidelines for determining how much of what they should take. And I realized that I'd had enough feedback from people, enough experience myself and interacting with enough people, that a step-by-step protocol had emerged that would allow anybody to move through it in order to set the dosage for themselves because we've also learned that there's a range of dosage.

So for many people who say, yeah, I tried one of each and it didn't work, it's like, okay, well, you need to do a little more work. The problem is we're all so different, both in the way we sleep, why we don't sleep, and how sensitive our bodies are to being pushed away from where they would normally be by supplements, that everybody needs a different dosage. And there's nothing you can do except work it out for yourself. So I provide that.

Leo: I ordered mine, got my oleamide.

Steve: Oh, good.

Leo: Yeah, so I'll try that now.

Steve: Good.

Leo: Does it come with a micro scoop?

Steve: Yeah, it does.

Leo: But you take 20 micro scoops' worth.

Steve: Yeah, yeah.

Leo: I should just use a tablespoon or something; right?

Steve: Well, and actually the page has a picture of the micro scoop. It shows that a micro scoop, a level micro scoop is 50 milligrams. But then you can get for a couple bucks standard measuring spoons. And so I've used them and weighed them to determine how much, how many milligrams of weight each different measuring spoon is. And so the one-eighth measuring spoon is 500 milligrams. And so I just, actually, I do two of those.

Leo: Oh, okay.

Steve: That makes it much easier.

Leo: That's a quarter, if my math may be correct.

Steve: Yeah.

Leo: Yeah.

Steve: So I also wanted to follow up on this what do we call a two-book series.

Leo: Yeah?

Steve: There was "trilogy," and you found the reference to "bilogy." Apparently "duology" is the formal noun for, as it described, a pair of related novels, plays, or movies.

Leo: Except it's not - it's incorrect Latin. It's not an existing Latin usage.

Steve: Right. And what I didn't explain is that it felt like there just must be a word that I had forgotten.

Leo: Right, but no.

Steve: Like trilogy is so obvious. We all know trilogy. What's the one for two? And it's like there really isn't one.

Leo: Isn't it funny?

Steve: Yeah. I don't get it. It's just like, okay. It's like, I must just be having, as they say these days, a brain fart. But no. There just isn't...

Leo: Isn't that weird?

Steve: There isn't a common word for two, and you would think there would be, one that would be as familiar to us as trilogy.

Leo: Apparently not.

Steve: No. A real quick note from Bob Port, who's in Albany, New York. The subject was "How SpinRite Kicks Butt." And everyone knows what the story is going to be. He said: "It finally happened. I knew it eventually would. A five-year-old hard drive" - okay, so not very old - "on a perfectly serviceable Dell Windows PC started acting up. It was just getting old, like all of us always are. I downloaded your utility, SpinRite, and voila. My hard drive was back in business, virtually good as new. How simple can life be? My eternal thanks. Bob Port."

Leo: Nice.

Steve: And so, Bob, thanks for sharing your success.

Leo: Voila.

Steve: Voila. And I found myself, when I was reading this, him talking about how simple it is, I've for years - many, many years - resisted adding this or that feature to SpinRite because, I mean, yeah, I could have taken the path of kitchen sink stuff. But what it does makes so much sense to just simply do what it does, which is just fix the disk. And in fact it's interesting because recently, when I was working with the beginning of the 6.1 project with a bunch of people online, I sort of suggested, what about if, in the next version of SpinRite I add the "beyond recall" feature? And the community pushed back hard and said, no, no, no, no, no, you know, make that a separate product, and we want SpinRite just to stay what SpinRite is. And it's like, oh, okay. So that's probably what I'll do.

And in fairness, my plan in the future is in fact to recognize that things change, with drives becoming so inexpensive now. I mean, it's just nothing to go get a replacement hard drive. SpinRite in-place recovery was born back when nobody had a second drive. I mean, you weren't using them as doorstops. They were really expensive. It was the most expensive component in the entire computer. And so SpinRite fixed the one you had. Seven will evolve to doing things like moving your data, being file system-aware and pulling files that specifically you ask - basically, in fact, adding all of those additional features that people have wanted over time. I'll have a good platform for doing that, then.

Leo: Nathan Hartley, our first question of the day today, Nathan Hartley tweeting: Listening to 578 and would rather see a push to IPv6 with encryption enabled kind of built in for everything than just this push towards IPv4 with HTTPS. Actually, that echoes what the Father of the Internet told me, what's his name. You know.

Steve: Famous guy.

Leo: You know that guy.

Steve: Yeah.

Leo: Father of the Internet, you know, Vint Cerf.

Steve: Vint Cerf. I'm glad you and I are of the same age, Leo. It's like, okay.

Leo: The guy. The dad.

Steve: Is he still alive?

Leo: Yeah, he's a beautiful guy, wonderful guy, beautiful guy, still alive. Made me cry. Because we had him on Triangulation. We had a two-part series. But I asked him, if you were doing it again - he did TCP/IP. I said, "If you were doing it again, what would you do different?" He said, "I'd put encryption in."

Steve: Yeah.

Leo: So this is a chance to go back and fix it.

Steve: Kind of. The problem is, and I get - I've had many people ask, so I was glad that Nathan's question hit me at a time when it was time for a Q&A. What was part of the spec of IPv6 was called IPsec, IP Security. And what it does is it formalizes, I mean, we already have IPsec, which can be used on IPv4, sort of as a higher level protocol. This puts it into layer three of the OSI seven-layer architecture. That is to say, IPv6 must support IPsec, as opposed to we're not sure if it's there or not.

But the problem is, and nothing shows this better than the ecosystem surrounding HTTPS and all of its challenges, IPsec doesn't solve any of that. That is, it's a mistake to just say, "Let's just put security in," because remember that what we're getting is two things. We're getting privacy from the encryption and authentication. And so the authentication means we're sure about who we're talking to. But that's a problem that IPsec doesn't itself solve because there isn't a solution, in the same way that HTTPS doesn't itself solve it. Those provide a carrier. They provide a possibility of solving it. They provide a substrate.

But you still need all of that other gunk which is so problematical. In the case of HTTPS we have certificate authorities and root trust stores and certificate chains and intermediate certificates and expiration and all that. So that's the best mechanism we've come up with for really employing HTTPS. IPsec is definitely useful for point-to-point sort of fixed connection applications. There is a good key exchange protocol. But it doesn't - it's not like just switching to IPv6 and suddenly everything is encrypted. And even if you were to employ, for example, opportunistic encryption, where you do create privacy, that still doesn't solve the authentication problem, meaning that a man in the middle could still decrypt and reencrypt unless you had end-to-end verification of who you were talking to.

And so I just wanted to - I had been looking for an opportunity to say, yeah, it's great that it's going to be there. But it's not like it's some miracle that allows us to avoid all of these other real bad or big and enduring and even getting worse problems that we see with HTTPS. All it does is it submerges it down into the lower level plumbing. And because it is bound into the spec, anything that says we support IPv6 inherently supports IPsec. Definitely good, so it's necessary; but it's not sufficient just by itself.

Leo: You know what we really need to do? We need to start a campaign. This is kind of off topic. But Google kind of did this HTTPS Everywhere campaign, and I think they've done probably a pretty good job of encouraging and Let's Encrypt supporting this HTTPS Everywhere. But, okay, that's good for security. But we need to do a campaign to end DDoS. And it's really a simple campaign, just to get ISPs all over the world to block traffic that's not coming from IP addresses they own. Right?

Steve: I wish that were the case.

Leo: Oh, but now with these amplification attacks does that mean...

Steve: No, no, because those are spoofed also.

Leo: Okay.

Steve: It's the in-band attacks. What the IoT devices are doing is issuing valid HTTP requests. And they don't care about being exposed. You know, some camera in someone's backyard...

Leo: Doesn't care.

Steve: Doesn't, you know. And so if they lose that little bot, you know...

Leo: That's the problem, isn't it, yeah.

Steve: Yeah. It's that unfortunately, just the IP spoofing, that would be part of a good solution. But what's happened is we've moved now to just flooding sites with valid queries.

Leo: Right.

Steve: Because now the devices...

Leo: You can block - there's so many IP addresses in there could be real because so what, you turn off that camera.

Steve: Right, and so...

Leo: There's 80,000 other ones.

Steve: Right. And so an ISP technically could see a ridiculous number of redundant queries. But that's a huge burden to place on them. They would then be doing traffic inspection in order to look at every single query leaving, and then making basically a value judgment. And, for example, we know it's impossible to come up with a good porn blocker because it tends to misfire. It misses things, and it blocks medical pages, for example.

Leo: Right, right.

Steve: So in the same fashion, it's just those sorts of heuristics just don't work.

Leo: Never mind.

Steve: But, yeah.

Leo: Never mind. Dante Bertana - oh, go ahead.

Steve: No, no, go ahead.

Leo: Question 2, another tweet. I wanted to let you know about DuckDNS, another free DDNS, or DynDNS. They even have copy/paste-able guides to configure your own Pi OpenVPN server: DuckDNS.org.

Steve: Yeah. A number of listeners have referred to them. I've heard they've got a great reputation. And of course we were talking about the idea of people running their own VPN endpoints where that made more sense than using a service. And so DuckDNS. I like that also, as Dante mentioned, they explicitly support the little Pi OpenVPN server, which is a simple $35 way of just sticking a VPN server onto your existing router. And then this solves the problem of its IP moving around if you're away from home. How do you find your IP if your ISP moves it? And Dynamic DNS does that. And DuckDNS provides it.

Leo: Is that DuckDuckGo? Is it the same people? Or is it another duck?

Steve: I don't think they're related.

Leo: Might be Aflac. Dagan Henderson: You have said in the past, Steve, that forcing users to change their passwords regularly offers no increase in security. In fact, we've even said it might even reduce security.

Steve: Yup.

Leo: Because they'd be tempted to reuse passwords or use variants of old passwords or just make them easy and short, easy to remember. But during Security Now! #578, you said rotating encryption keys (DNSSEC) is considered a good security practice. In both cases, isn't it just a secret being expired and replaced? Why is key cycling a good practice, while password cycling is unnecessary?

Steve: So a great question. And as your comments, Leo, indicated, it is gray. There is no question. We know from experience that where corporations have forced their entire employee base to routinely change passwords, there is a huge back pressure from the people who the IT department are imposing this upon, to the extent that they'll, for example, game the system. They'll figure out that, oh, look, if I rotate among three, that fools it because it only keeps track of the previous two. And so I give it a third, and that pushes the one that I'm giving it off the end of the little queue. Or they'll, like, deliberately change it six times right away and then right back to the one they originally had. I mean, they just - they don't want to do it.

And so in the case of DNSSEC, though, it wasn't just for the sake of rotating keys. It was also the fact that they were going from a 1024-bit to a 2048-bit key. And so that's a ridiculously large improvement of security. And it's the kind of thing you don't, you explicitly don't want to have to do very often. So their feeling was, since we're not in an emergency, everything's kind of quiet right now, we can schedule it, people can buy airfare and so forth for the meetings to come together. Now would be a nice quiet time to do it.

And the other reason is we've talked about the crazy security that that process is surrounded by. Well, that also says you don't want to do it very often. It is the weak time when all that is happening. When this key change is occurring, that's when the system is most vulnerable. So you want to do it infrequently. But then that also means you want to get as much value from that as you can. You want to get enough bang for the effort.

So again, there isn't any sort of black-and-white. I do argue that changing keys seems more like a bad idea than a good idea because the only argument that I can see for it is if there's a reason to believe old keys are compromised. And that's when we are changing passwords; right? When a service announces, oh, shoot, we just lost half a billion of our subscriber accounts' passwords, everybody has to change their password. So, okay. There is makes sense. So what some people have felt is, okay, maybe we don't know of breaches which have occurred. So we're just going to preemptively force everyone to change their password. Wow, you know, that's just - that's an expensive thing to ask a large community to do. So it's a gray area, but I think it could easily be overdone.

Leo: Question, let's see here, 4 from a fellow named Steve. Hmm.

Steve: Not me.

Leo: Is that his real name? Hey, Steve. Just finished your latest podcast, still a highlight of my week. A quick note about Windows 7 update problems. I'm in a team that administers 100s of Windows 7 machines and have found that, if a Windows 7 machine has SP1 installed, simply stopping the update service, then installing Microsoft KB3102810 does the job every time. We run WSUS servers, and heaps of machines were not updating. The job was given to me. I made a group policy startup entry. It's just a batch file that checks to make sure Windows 7 is running, checks the architecture, then downloads the appropriate update. Windows actually has a built-in wget equivalent called bitsadmin. So the batch file stops Win update service, installs the update. It can still take a couple hours for the machine to finally finish searching for updates the first time, but it has had a 100% success rate so far.

Steve: I really appreciate...

Leo: What does that update do?

Steve: So that's the one that informs Windows of the change in the update service that Microsoft offers. So if you get a brand new, you know, Service Pack 1 was the one and only service pack for Windows 7. So if you get a Windows 7 machine with SP1, which is the last image Microsoft made available, and you just say, you know, you install it, and you say go, it just, well, as he said, and as all of our listeners who've tried this know, it just - oftentimes days go by, and nothing happens.

Well, it turns out that, since SP1, Microsoft changed the actual service. And so the Windows Update back at the time of SP1 doesn't know how to contact Microsoft. The API was not done in a backward-compatible fashion. So you stop the update service, install just this one update, and that then informs Windows 7 how to get the other updates from Microsoft. But the key here, and this is what's often missed, is manually stopping the update service. That's the key. Most people just sort of assume it won't be in the way, and so they try to install this update while the update service is running, and there is a collision. So you stop the update service. Then you install this and then, like, reboot your system. Update service will restart, and then it will move forward.

Leo: Kind of weird, Microsoft doesn't say anything about stopping it in the note about this.

Steve: No.

Leo: Although they mention that svchost.exe, one of the symptoms is occupying 100% of CPU usage. Is that the update service, svchost.exe? I would guess that is, huh?

Steve: Yeah. Yeah, the svchost is a generic process that hosts a whole bunch of different services.

Leo: Okay. So stop the update service, you know, you know how to do that; right? And then apply this patch.

Steve: Yeah. If you go under the Admin tab, there's like a Services. Or you can click on - I'll never forget, it was Crazy Chris.

Leo: Pirillo.

Steve: Pirillo, yeah, exactly.

Leo: All you had to say was Crazy Chris. I know who you're talking about.

Steve: Crazy Chris; right. It was he, because I was new to whatever version of Windows it was, must have been XP, and he said, "You right-click on My Computer."

Leo: My Computer. Select Properties, yeah.

Steve: Well, actually it's...

Leo: Actually it's system management; right? Yeah.

Steve: It's Manage. You select Manage, and that brings up that little tree. And among there is Services. And then as you scroll down you'll see Windows Update Service. And so you just click the little stop button, and that stops it, and then install this update, yeah.

Leo: Nice, very good.

Steve: But it was Chris Pirillo who - and I thought, hey, I didn't know you could get to it that way. Very nice.

Leo: You can also, I think, Windows - I wonder if Windows X works on Windows 7? I think it might. Windows Key X is the way to do it on Windows 10. It gives you a great menu of stuff.

Question 5, Good Morning: Good morning, Steve. In your episode "Flip Feng Shui" you mentioned that researchers flipped a bit in a public key, making it easily factored to find the private key. The problem is, if you flip a bit in a public key, it's no longer a product of the private key. How would you then figure out what the private key was? So that's a good question.

Steve: It's a great question. And in my description of how this worked I failed to explicitly note that what the Flip Feng Shui guys had done - I explained how they deliberately caused the memory coalescence so that their process was sharing a memory page with the target process, and that they were then using the Rowhammer attack to flip a bit in where they knew was the private key. Okay, but what I didn't explain was that what that does is it flips the bit in the one copy that they both share. So in doing that, you have changed, as Mr. Good Morning here notes, you've changed the key in the other process. So then in the attacking process they perform the factorization of the modified key. They then extract the private key and immediately use it, like right now, to log into the process being attacked using their private key. And it works because it's now being compared against the modified shared public key. And so in my description I just sort of - I left out that final piece.

Leo: Wish you had told me. I've been trying to do this Flip Feng Shui all week. Joey Bueno. Hey, Joey. Steve, could you check out Intel Security's True Key app? It's free, and it leaves LastPass for dead. My question is, can we trust it? Could you please check it out and let the world know if it's okay to use it? Or is it a spy?

Steve: So, okay. So I thought, okay, this is interesting. I should find out what Intel's True Key is. So I dug in, and I didn't see anything special about it. It's free only for 15 passwords.

Leo: Oh.

Steve: Yeah, exactly. Thank you.

Leo: What are you going to do with that?

Steve: Yeah. Which is useful only as a trial, of course.

Leo: Right.

Steve: So not useful for any real use. Any real use costs $20 per year, with an apparent maximum of 2,000 passwords.

Leo: What?

Steve: Well, okay, so that's a lot, but why have a maximum?

Leo: Right.

Steve: But that's what they say. The only hook that this has for appeal appears to be Intel's biometrics, which is really, you know, that's what they're actually trying to leverage is their camera and their scanners and so forth. And so that's okay. But biometrics are already built into mobile devices and will soon become standard options and features on probably everywhere. So, for example, by comparison, LastPass and even the SQRL Client for iOS already reprompt the user for their fingerprint confirmation when you use LastPass or SQRL. So Apple, as we know, made the API available so that apps could ask you to reassert your identity. I know, for example, the Amazon app on iOS does that, too, which is nice because sometimes it asks, like, twice. And I have a Pad that doesn't have the fingerprint reader, and now they're making me type in my password twice. It's, like, so annoying. But, yeah.

Leo: Fingerprint's awesome. Let's get everybody - let's just, you know, just - we all have a phone now. Almost everybody should have a phone by now with a fingerprint reader on it.

Steve: Yup.

Leo: Apple with OS X Sierra now lets you unlock your Mac using the fingerprint reader on your phone.

Steve: Nice. We're getting there. We're getting there.

Leo: A little Catch-22, though. I thought, oh, great, I'll set that up. And then it says, well, you have to have two-factor authentication on, on your iCloud account. I said, oh, that's great. I didn't even know they had that. They added that last version, El Capitan. So I was going to turn on two-factor authentication. It said, okay, but we've got your security questions right here. What was the name of your best friend in high school? I don't - I must have set these questions up seven years ago.

Steve: And, if you were smart, you probably...

Leo: You didn't use the real answer anyway.

Steve: Exactly. You used Joey Bueno.

Leo: I have no - I said Joey Bueno was. I remember him well. So I had no idea. So I call Apple to reset my security questions. And by the way, once you turn on two-factor, your security questions go away, another good reason to use two-factor because security questions are the worst idea in the history of security.

Steve: Yup.

Leo: So she reset my security questions. And I go back, and it says, well, since you've changed your account, you can't turn on two-factor. First it said for three days. I understand that. Three days would be fine. I understand that.

Steve: Yup.

Leo: Then it said 30 days. Three days is so a hacker can't just kind of instantaneously reset all your stuff, give you a little time to notice. Thirty days? Really, Apple? So now I'm completely insecure for 30 days?

Steve: Wow.

Leo: Wow. Funvacuum, not his real name. Maybe it is. I don't know. Shouldn't say. He might be Finnish. SQRL. SQRL, Steve. You may have already covered this. I'm on Episode 564. I'm a little behind. Do you store a list of websites so we can automatically rekey for all sites?

Steve: That question comes up a lot. And the answer is I don't. It's not part of the spec. Okay. So what he's asking is, as we know, with SQRL you are able to securely rekey your existing SQRL key if you have some reason to believe it's been compromised. So that's part of SQRL's full lifecycle key management. But it's incumbent upon you to go to the various sites and do that. And you can lock them to keep a bad guy from logging in as you until you're able to rekey. That's easy to do. Or with your rescue code you are able to rekey and then prove your identity which a bad guy is unable to do in order to get control back. So this all works. It's been proposed, and I'm not - I have not put it into my client for a couple reasons. If it were there, then a bad guy who got your key would also get the list of all the sites where you have used it, and where you visit, and where he should go attack you. And that seems like a bad idea.

Maybe it could be encrypted, I mean, if I had to solve the problem, I could. I could come up with a way where it would be dynamically stored, and only the rescue code could be used to decrypt it. Maybe somebody will want to do that. I sort of figure, first of all, the system is designed for you not to lose your key, and this is a secure recovery system. And all you have to do is go to the most important sites first and either shut them down until you get a new key or change the key. And then you can catch up with the less important ones over time. Basically, SQRL always carries your old and your new key and is able to seamlessly update them. So it's really pretty easy. But I just don't - I don't really like the idea of trying to hold onto every site you've ever visited because it is a potential attack vector.

Leo: Christian Turri asks: LastPass is now showing an alert if you have a Yahoo password in your LastPass vault. That's very nice. This prompted me to run the LastPass security challenge, which I'm sure you know what it is. Once the scan on all my passwords was finished, it showed a section of compromised accounts I should change passwords for. Among them were Yahoo and Dropbox, of course totally expected. I was surprised to see my Amazon account there, too. Hadn't heard of any attacks on Amazon. Appears they have also had and informed some customers about it. I hadn't heard that, either.

Steve: Yeah, I just wanted to share that. I love the fact that LastPass is doing that. And I would remind people from time to time, run that little security test that LastPass offers. You'll probably be a little bit surprised. And in this case, surprise is a good thing.

Leo: LastPass makes it easy, though, to fix the duplicated passwords, the bad passwords, the change of passwords.

Steve: Yeah, yeah.

Leo: And just a follow-up, Joe Siegrist is still doing a great job there. The LogMeIn acquisition doesn't seem to have hurt LastPass. In fact, I think we have seen evidence of improvements. I think it's still usable; right?

Steve: Yup. I'm staying with it.

Leo: Yeah, me, too. We use it for Enterprise, so we're kind of all in on it. Okay. Three fun ones. You ready?

Steve: These are quickies. And then we're done.

Leo: MSPoweruser: Microsoft adds 24-bit color support to the Windows Console. What? The c-prompt?

Steve: I love this. This is so good. This is - and the show notes have a picture of the color palette that you can now present in the Windows Console. And to me this is, okay, we have run out...

Leo: Of things to do.

Steve: ...of actually useful things to do.

Leo: In their defense, it probably wasn't that hard to add; right?

Steve: What can we, you know, we've got to have a new version. Oh, my god. That's what we need. We need 24-bit color for the command prompt. Thank you very much.

Leo: And here's a tweet from Mikko Hypponen. Of course we know him. We had him on Triangulation. He's a security guy...

Steve: I love this.

Leo: ...at F-Secure. He tweets: Yahoo's ad revenue is skyrocketing, as 500 million users log into Yahoo for the first time in years to change their password and log out.

True. True.

Steve: Yes. How many eyeballs have seen those annoying ads? Yes.

Leo: I turned on two-factor, too, because they have that.

Steve: Okay, and - okay.

Leo: I was just going to say - we talked about this last week. Better to do that than to delete your Yahoo account because you don't want somebody reusing your email.

Steve: Good point. And we did cover the problem of email account reuse.

Leo: Right.

Steve: Somebody was deleting them a long time ago, and it caused a lot of problems.

Leo: Yahoo was.

Steve: Yahoo was.

Leo: Ironically, Yahoo announced that, if you don't use your email for a period of time, if you let it lie fallow, we'll recover your email address and use it again. So, especially if you have an email address like I do, that's yourname@yahoo.com, you might want to just kind of keep it around. And every few months I'm going to send and receive an email there so they think I'm still using it.

Steve: Every mistake possible.

Leo: Couldn't be worse. Couldn't be worse.

Steve: And this last one, Leo, I just love it because it shows us that we are in fact getting up there in age.

Leo: Oh, my god. This must be a - oh, well, I'll just read it. An anonymous listener says: Here's an esoteric question. Do you know of any audio-based protocols or standards for transmitting ASCII symbols? Did old machines beep and chirp? Wow.

Steve: Isn't that great?

Leo: Do you think he's joking, or he's serious?

Steve: I think he's serious. And I just love it. He probably doesn't have to shave yet.

Leo: Doesn't know what dit dit dit dot dot dot dit dit dit means?

Steve: Or what's a modem?

Leo: Or a modem.

Steve: Alex.

Leo: Yeah.

Steve: Yeah. It's just - I just loved it, you know, an audio-based protocol or standard for transmitting ASCII symbols.

Leo: There must be a way.

Steve: Yeah, there are a few of those.

Leo: But even going back to Samuel F. B. Morse, and they had those wires strung up along the Pony Express routes. And he said, you know, we should really have some way of sending messages over those wires.

Steve: Yup.

Leo: And we'll make a dit dit dit be an "S," and a dah dah dah be an "O."

Steve: Dit dah.

Leo: Dit dah.

Steve: Yup. I think that's "A"; isn't it?

Leo: Yeah, the ETAIO, you know, the first most used letters are the simplest.

Steve: Yes, which was brilliant.

Leo: Brilliant.

Steve: And basically it's sort of a form of Huffman coding. The ones you use most often are the shortest.

Leo: Yeah. His story is interesting. You know he was a painter, a very, very, very good painter, shown in the salons of Europe and France. If you ever want to read...

Steve: Morse?

Leo: Morse.

Steve: Really. Interesting.

Leo: Well, you know, I only realize that when I'm looking at these great paintings from Samuel F. B. Morse and thinking, gee, I wonder if there's any relation to Morse code?

Steve: And did he do anything else of note?

Leo: I think so. I think he had quite the life.

Steve: Oh, cool.

Leo: I don't - there must be a biography of Morse somewhere. But, yeah, if you just look in the Wikipedia, you'll find some interesting stuff. That's our assignment for this week. Are you not doing any brain teasers anymore? Or they just don't - they're hard to come by?

Steve: Yeah. If any present themselves, I absolutely will.

Leo: Those were fun. I liked those.

Steve: Yeah, those were great. I have...

Leo: That would have been a good one. Does anybody know of any way to transmit ASCII characters in sound?

Steve: Yeah. Can you imagine the amount of effort and time that has gone into that over the years, you know, the sound of geese mating is what I always called the [crosstalk] x-dot...

Leo: [Modem sounds]

Steve: Yeah, exactly. Boing boing boing.

Leo: [Modem sounds] Is there a direct, I mean, I guess it's bits; right? [Modem sounds] is ones and zeroes; right?

Steve: Yeah, yeah. Oh, I mean, it started as two frequencies, of course. You know, frequency-shift keying, FSK. And then, as they kept pushing the standard further and further along, you got all kinds of fancy quadrature and much, much more complex modulation of audio, to the point where, again, as an old-timer, it's like, you can't put 56K through a phone line. What are you talking about?

Leo: Actually, there is - this would be a good - here we go. Here's your brain teaser.

Steve: Okay.

Leo: Is there a theoretical limit to the amount of data you can push down an analog copper wire? And we'll be back next week, and Leo will do some research. Because I have a feeling I'm going to have to. There is, actually.

Steve: I know the topic for next week.

Leo: Yes?

Steve: But I am embargoed.

Leo: [Gasp]

Steve: Yes.

Leo: So a surprise.

Steve: So we have a big podcast next week.

Leo: Can't wait.

Steve: Yeah. It's going to be a goodie.

Leo: Can't wait. Well, here's the deal. Come back here Tuesday, 1:30 Pacific, 4:30 Eastern in the P of M, post-meridian. Or, for those of you who are on a 24-hour clock, 20:30 UTC. And you will find me and Steve jawing about security and all that stuff. You can also join us in the chatroom, irc.twit.tv as you watch, or be in-studio, too, like Nick did. He came up from Menlo Park to join us. He's a fan. Hi, Nick.

Steve: Oh, cool.

Leo: Yeah. Email tickets@twit.tv, we'll put a seat out for you. And a pair of headphones. And you can also, of course, watch after the fact. We make on-demand audio and video available. Steve's got the MP3 plus transcripts, which is really nice, available at his site, GRC.com. And while you're there, don't forget to pick up SpinRite, the world's best hard drive recovery and maintenance utility.

Steve: As the slogan for it says: "It works."

Leo: What is the slogan? Oh, it works.

Steve: It works.

Leo: It works. It's a good slogan. Pithy. He has lots of other free stuff there, including his Healthy Sleep Formula and Perfect Paper Passwords. And you probably first discovered Steve, as many did, with ShieldsUP!, which is still there testing your shields, your router.

Steve: Yeah, as this whole IoT thing has happened, there's been a real resurgence in people wanting to make sure, like, what's going on with their homes.

Leo: Is my router working, yeah.

Steve: Yeah.

Leo: We also have audio and video at our site, TWiT.tv/sn for Security Now!. And you could find the show as a podcast, so you can find it everywhere podcasts are aggregated, like iTunes and Stitcher and Slacker and Google and everywhere else. You know, we also have an Alexa, I'm sorry, an Amazon Echo briefing. You know, we're on the on the flash briefing - Steve sometimes is, as well - a daily tech news headline. If you ever do your Amazon Echo flash briefing, what you should do is go on the app and search for TWiT and add us to it. It's not automated. It's real voices. Usually it's Jason Howell and Megan Morrone from TNT. But we also do bits from the other shows and so forth. That's for your Amazon Echo. Thank you, Steven.

Steve: You know, you think about ShieldsUP!, I was never able to scan more than that block of service ports because I developed it all when I was behind my two T1s. But I'm at Level 3 now. And it might be time...

Leo: You could do more.

Steve: It might be time to do them all.

Leo: All 65,636?

Steve: Yeah.

Leo: Hmm. Why not? I didn't know you didn't do them all. You just do the canonical zero through 1024?

Steve: Actually, I go one line above. I think it's - I don't remember what the last port number is. But I go a little bit past that because there was some reason, once upon a time.

Leo: To cover some port.

Steve: To do that.

Leo: Some port up there.

Steve: And then I do the standard port scan or the full port scan. But those are only the service ports which, back in the day, those were the ones where you would be running services. But these days and age you're going to have a camera up at port 50,000.

Leo: Oh, I have stuff on all sorts of ports. Yeah.

Steve: Yeah. Ah. Don't worry, everybody. I'm going to get SQRL finished first, and I'm going to get on...

Leo: Yeah, you're going to make people mad.

Steve: Yeah. They are, oh, yeah.

Leo: You have a few things on your plate. You have a punch list now, Steve.

Steve: I do. But the good news is it's fully parameterized. When I rewrote it, it's just like, I could just tweak it a little bit and suddenly be scanning the entire port range.

Leo: See, that's smart.

Steve: And now I have the bandwidth, which I didn't used to have.

Leo: That's smart.

Steve: But, no, I won't do that soon, I promise. I'll get other stuff done first.

Leo: You won't be allowed.

Steve: No.

Leo: Thank you, Steve. You work hard enough on this show, by the way. Thank you for that hard work. We really appreciate it.

Steve: My pleasure, my friend. Talk to you next week with a big story.