Are Passwords Immortal?

Description: This week we cover the action during last week's Pwn2Own Mobile hacking contest. As this year draws to a close, we delve into the final last word on processor misdesign. We offer a very workable solution for unsupported Intel firmware upgrades for hostile environments. We look at a forthcoming Firefox breach alert feature. We cover the expected takeover of exposed Docker-offering servers. We note the recently announced successor to recently ratified HTTP/2. We cover a piece of 1.1.1.1 errata, close the loop with some of our podcast listeners, then finish by considering the future of passwords using a thoughtful article written by Troy Hunt, a well-known Internet security figure and the creator of the popular HaveIBeenPwned web service, among others.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-690.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-690-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson's here. And, boy, do we have a big show for you. He's going to talk about the most recent Pwn2Own and which operating systems fell to hackers' attention; the source behind all of these Spectre/Meltdown and other Intel processor woes and why it's going to be a tough one to fix. And then he's going to talk about passwords and the future of passwords. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 690, recorded Tuesday, November 20th, 2018: Are Passwords Immortal?

It's time for Security Now!, the show where we cover your privacy and security online with the guy in charge, Steven Gibson of GRC.com. Hello, Steven.

Steve Gibson: Hey, Leo. It's great to be with you again for 690.

Leo: Yikes.

Steve: Six nine zero, for this, what is it, two days before Thanksgiving in 2018. I guess this is, what, this is an earlier Thanksgiving than usual? Is that what happened? I saw something like on that.

Leo: I guess it is, yeah.

Steve: What's it supposed to be, the third Thursday of the month?

Leo: Right. So the earliest it could be would be the 21st, I guess.

Steve: So this is certainly close to that.

Leo: Yeah.

Steve: Our topic is, or the title of the podcast, asks the question, "Are Passwords Immortal?" That is, will they never die? And this comes from an article, or this was inspired by an article that a friend of the podcast, Troy Hunt, wrote 15 days ago.

Leo: He's HaveIBeenPwned. He's the HaveIBeenPwned guy.

Steve: Yes, he is, exactly. He's the HaveIBeenPwned guy who offers the service that allows people to put their email addresses into his site confidentially. And Troy gathers all of the breached data from websites that lose control of their data all over the Internet and lets you know if your email is among those. So anyway, so that's the main topic. There wasn't a ton of news. So we have time to go into that in some detail, and I plan to.

I reached another milestone in my proposal for what may ultimately put passwords to rest. We'll see. And I'm going to be a little more philosophical about how I feel about that because I don't disagree with the position that Troy has. I'm going to share what he wrote at the end of this podcast, and then we'll talk about it and also put it into this context.

But we also have, as we anticipated, last week was the 2018 Mobile Pwn2Own hacking contest, which took place in Tokyo. And we talked - there was a - somebody had done a zero-day which we covered a few weeks before that, but that was for a desktop platform. And he mentioned in his Twitter feed that he'd be going to this one. So we have the news from that.

Also, as this painful year of processor design issues winds to a close, we have the ultimate final last word on processor misdesign. We teased the forthcoming of this research a week or two ago. It has now been published, and it is amazing and comprehensive, and it's got some surprises, of course, because a bunch of academics really wrestled this whole topic to the ground once and for all.

And there's some very good news. There's a workaround available for anyone whose motherboard BIOS is not updated, has not been updated, will probably never be updated. If they're concerned about not having the latest firmware for their own processor, we've got a way around that now. Which is actually probably very practical for server environments, which as we know is where the danger is greatest in virtualized environments where the chance of hosting hostile code is greater.

As we've been saying all year, yeah, it's not good to have Spectre and Meltdown vulnerabilities, but for the typical end user it's not the end of the world because, if something has already gotten into your system, you've sort of already lost that game. So anyway, we'll talk about this really cool solution for that.

We've got a forthcoming Firefox breach alert feature that'll be joining Firefox browsers. We will cover the expected takeover of exposed Docker service offering servers, which again we expected, and it has happened. Also we will note the recently announced next version or successor to the only just recently ratified HTTP/2. We now know what HTTP/3 will look like and be. Also I want to cover a little bit of what I will put it in the errata sector about this 1.1.1.1 that we stumbled over on the fly last week when I was surprised by suddenly realizing that, wait a minute, https://1.1.1.1 was giving us an extended validation secure connection. And it's like, what? How did that happen? So of course I know how. And even John Graham-Cumming dropped a piece of email to me saying, "Hey, Steve, if you're puzzled, I can explain it to you." And by that time I knew. I just had to think about it a little bit more.

Also, as I said, we'll have a chance to close the loop with some of our podcast listeners, some little bits of confusion that exist, and some other points that are useful. And then we're going to take up this issue of the future of passwords as a consequence of this thoughtful article which you had already encountered. And you can imagine, I was bombarded by it from every direction. And what's interesting is the very first response to Troy's posting was someone saying, "Have you heard about SQRL?" Anyway, and then a lot of the dialogue in the conversation thread that ensued was about that. So I think a great podcast for our listeners. And I couldn't resist our Picture of the Week, just become someone happened to send it to me, and I thought it was fun. So, yeah. Good stuff.

Leo: Yes, I'm very excited.

Steve: So last week was, as I mentioned at the top of the show, November 13th and 14th was the two-day Pwn2Own, which is an annual, well, actually it's probably more than that because it occurs typically...

Leo: This is a mobile one.

Steve: Right. And it occurs sort of in concert with various security conferences where people are gathered together. Trend Micro in this case was behind this one, offering cash and prizes during this competition for vulnerabilities and exploitation techniques against the listed devices. We talked about those a week or two ago, basically handsets, a couple IoT cameras and so forth. Those didn't get - oh, and also the Apple Watch was on the list. Nobody attacked those.

The write-up is fun because it sort of takes the sports announcer style, which I'll share with our listeners. Trend Micro wrote of each day. They said: "The first day of Pwn2Own Tokyo 2018 has come to a close. Today saw some great action and amazing research as we awarded $225,000 USD and 48 Master of Pwn points. We had six successful exploits and purchased 13 bugs in total."

They say: "Our day began with Fluoroacetate" - which is the team of Amat Cama and Richard Zhu - "successfully exploiting the Xiaomi Mi 6 handset via NFC. Using the touch-to-connect feature, they forced the phone to open the web browser and navigate to their specially crafted webpage. During the demonstration, we didn't even realize that action was occurring until it was too late. In other words, a user would have no chance to prevent this action from happening in the real world. The web page exploited an out-of-bands write in WebAssembly to get code execution on the phone. This earned them $30,000 and six Master of Pwn points.

"The Fluoroacetate duo returned, this time targeting the Samsung Galaxy S9. They made quick work of it by using a heap overflow in the baseband component" - that's the cellular radio component - "a heap overflow in the baseband component to get code execution." They write: "Baseband attacks are especially concerning, since someone can choose not to join a Wi-Fi network, but they have no such control when connecting to baseband. The work earned them another $50,000 and 15 more points towards Master of Pwn.

"Next up, Amat and Richard returned to the Short Distance category. This time, they were targeting the iPhone X over Wi-Fi. They used a pair of bugs, a JIT" - that's a just-in-time compilation - "a JIT vulnerability in the web browser followed by an out-of-bands write for the sandbox escape and escalation." Okay, this is from an iPhone X. "The successful demonstration earned them $60,000 more, and 10 additional Master of Pwn points." They said: "This ends the first day of competition with $140,000 and a commanding lead for the Master of Pwn with 31 points.

"Following that attempt, the team from MWR Labs combined three different bugs to successfully exploit the Samsung Galaxy S9 over Wi-Fi. They forced the phone to a captive portal without any user interaction, then used an unsafe redirect and an unsafe application load to install their custom app. Although their first attempt failed, they nailed it on their second try to earn $30,000 and six more Master of Pwn points."

They said: "In our last entry of the day, Michael Contreras made sure his first Pwn2Own attempt was memorable. He wasted no time in exploiting a type confusion in JavaScript. In doing so, he earned himself $25,000 and six Master of Pwn points. We look forward to seeing him in future events." And they wrote: "Excelsior!"

Then they start the second day: "Our second day began with the Fluoroacetate duo of Amat and Richard targeting the iPhone X in the browser category. After a stellar first day, they kicked off Day Two in style by combining a JIT bug in the browser along with an out-of-bounds access to exfiltrate data from the phone. For their demonstration, the data they chose happened to be a deleted picture, which certainly was a surprise to the person in the picture. The research earned them $50,000 and eight more points towards Master of Pwn.

"Next up, MWR Labs team of Georgi Geshev, Fabi Beterke, and Rob Miller also targeted the iPhone X in the Browser category. Unfortunately, they couldn't get their exploit chain to work within the time allotted, resulting in a failed attempt. However, they did have some great research, and we acquired," writes Trend Micro, "the bugs through our normal ZDI program." That's the Trend Micro Zero-Day Initiative program.

"Following that, the Fluoroacetate team returned, this time targeting the web browser on the Xiaomi" - is it Xiaomi? I always say it wrong.

Leo: Xiaomi. Xiaomi.

Steve: Xiaomi, thank you, "Xiaomi Mi 6. They continued their successful contest by using an integer overflow in the JavaScript engine to exfiltrate a picture from the phone. They earned themselves another $25,000 and six Master of Pwn points. The Fluoroacetate team couldn't keep their momentum going throughout the entire competition, as their last entry fizzled out. They attempted to exploit the baseband on the iPhone X, but could not get their exploit working in the time allotted."

Leo: To clarify, the baseband is the radio; right? The SS7 radio software.

Steve: Yes, correct. Yeah, the cellular radio. And we've often talked about how all of the attention is being focused up at the application processor, where the apps run. But very little attention gets given to this whole separate processor, the baseband, which does all of the cellular stuff. And it's believed to be riddled with bugs, which haven't really been looked at yet. So by offering prizes for exploits down there, we're beginning to see those surface.

So they said: "Still, five out of six successful demonstrations is pretty remarkable," speaking of the Fluoroacetate team. They said: "We're glad to see these two team up and hope to see more from them in the future.

"Our final entry for this year's event saw the MWR Labs team target the web browser on the Xiaomi Mi 6 handset. The team combined a download bug along with a silent app install to load their custom app and exfiltrate pictures. This earned them another $25,000 and six additional points toward Master of Pwn."

They write: "This closes out Pwn2Own Tokyo for 2018. With 45 points and $215,000, we're happy to announce the Fluoroacetate duo of Amat Cama and Richard Zhu have earned the title Master of Pwn. Overall," they write, "we awarded $325,000 over the two-day contest, purchasing 18 zero-day exploits. Onsite vendors have received the details of these bugs and now have 90 days to produce security patches to address the bugs we reported. Once these are made public, stay tuned to this blog for more details about some of the best and most interesting bugs we saw this week."

So this is a nice way to do this. You produce meaningful prizes, meaning money, to incentivize talented developers, talented hackers, to dig in and find these problems. As a third party, in this case Trend Micro is a commercial entity, so they get contributions from others, pool the funds. They form this in a contest. The hackers do the best they can in a competitive mode. So they both want the crown of Master of Pwn, and they certainly want, you know, not bad to come away with $215,000 for a team. We don't know how long it took them to get these. And then 18 previously unknown, I mean, like no one knew these problems existed, 18 serious flaws in mobile handsets are then responsibly disclosed to their vendors under the onus that 90 days from then these will go public. And as a consequence, everybody gets protected.

Leo: You know what fluoroacetate is?

Steve: Fluoroacetate, no.

Leo: It's bug poison.

Steve: Ooh. Clever.

Leo: Isn't that?

Steve: Very clever. Very clever.

Leo: Sodium fluoroacetate or potassium fluoroacetate.

Steve: Bugs don't like it.

Leo: Bugs don't like it.

Steve: Cool. That's neat.

Leo: I know. I had to look it up once you said it. I thought, I bet you there's a reason they call themselves that.

Steve: Okay. So anyway, I just wanted to finish and say, you know, compared to the tweets we've been covering recently from a couple developers who are coming up with good, beautifully engineered exploits that they're just dropping out onto the world, saying here you go, I mean, anybody who wants it can have it, I just - I don't get why they're not willing to operate in a responsible fashion and make some money. I mean, ultimately that puts bread on the table.

Okay. So a couple weeks ago, we knew that there were - I think the number was seven, seven previously unknown - everybody was like, oh, you've got to be kidding me - Spectre-ish/Meltdown-ish new problems that a research team had found. At that point, that's all we knew. We didn't have the details. Now we do. This picture on this next page of the show notes is a tree diagram which shows the consequence of an academic, methodical, careful analysis of contemporary processor design. Those red boxes are demonstrated new attacks. The dotted ones were sort of theoretical maybe that they were not able to actually do, but the red ones actually happened.

I want to share the abstract and the beginning of the introduction of this because, more than anybody else, these guys - I wish we had this a year ago because things would be very different if we had this a year ago. I have to wonder what's going on with Intel because they felt like they were reactive, and we know that there was some understandable CYA behavior with their PR people and all processors have these problems and so forth. And we've talked about that honestly, the fact that, yes, many things have been done without a proper focus on security. What annoys me is that ultimately we rely on Intel to be the designers of secure hardware. And even now, a year later, as we're about to hear, these problems have not been fixed.

So the abstract of this paper, which more perfectly frames this than I've seen anywhere else, reads: "Modern processor optimizations such as branch prediction and out-of-order execution are crucial for performance. Recent research on transient execution attacks, including Spectre and Meltdown" - by the way, I should say "transient execution attacks," that's their term for the umbrella that they've created for understanding these. "Recent research on transient execution attacks, including Spectre and Meltdown, showed, however, that execution or branch misprediction events may leave secret-dependent traces in the CPU's microarchitectural state. This observation led to a proliferation of new Spectre and Meltdown attack variants and even more ad hoc defenses, for example, microcode and software patches."

They write: "Unfortunately, both the industry and academia are now focusing on finding efficient defenses that mostly address only one specific variation or exploitation methodology. This is highly problematic, as the state of the art provides only limited insight on residual attack surface and the completeness of the proposed defenses.

"In this paper we present a sound and extensible systemization of transient execution attacks. Our systemization uncovers seven new transient execution attacks that have

been overlooked and not been investigated so far. This includes two new Meltdown variants," which they named Meltdown-PK on Intel and Meltdown-BR on Intel and AMD. "It also includes five new Spectre mistraining strategies. We evaluate all seven attacks in proof-of-concept implementations on three major processor vendors: Intel, AMD, and ARM. Our systematization does not only yield a complete picture of the attack surface, but also allows a systematic evaluation of defenses. Through this systematic evaluation, we discover that we can still mount transient execution attacks that are supposed to be mitigated by rolled out patches." Meaning this stuff hasn't been fixed yet.

So to get into it a little bit deeper, in their introduction they say: "Processor performance over the last decades has continuously improved by shrinking processing technology and increasing clock frequencies, but physical limitations are already hindering this approach. To increase the performance, vendors shifted the focus to increasing the number of cores and optimizing the instruction pipeline. Modern processors have deep pipelines, allowing operations to be performed in parallel in different pipeline stages or different units of the execution stage. Many processors additionally have a mechanism which allows executing instructions not only in parallel, but even out of order. These processors have a reordering element, which keeps track of all instructions and commits them in order. In other words, there is no functional difference to regular in-order execution if they have a dependency on a previous instruction which has not yet been executed and committed. Hence, to keep the pipeline full at all times, it is essential to predict the control flow, data dependencies, and possibly even the actual data."

Now, I'll just pause and say for a minute that the idea of designing that just - it is just mindboggling when you consider that you have interdependent instructions which are themselves very complicated, which are each producing outcomes, which subsequent instructions may or may not be dependent upon. And you're basically wanting to run ahead as far as you can while slower things are taking longer to execute or having to go fetch things that are not in one of three layers of caches out of main memory, to then get the result, and then that may cause a branch decision to be made which happened in your history.

And then it's like, oh, we're going to have to unwind all of this because we guessed wrong. Or, oops, wait, we've got to stall this because now we're waiting on a result from something, I mean, I just can't even imagine. This is in hardware. Or, well, firmware, with the aid of microcode. But, I mean, wow. So just reading that description of today's contemporary microprocessor, for one thing it makes me feel like it's worth the money. A few hundred dollars for that technology? Yeah.

Leo: Good point.

Steve: Whoa. So they said: "Flushed instructions, those whose results are not made visible to the architectural level due to roll-back, are called 'transient instructions.' On modern processors, virtually any instruction can raise a fault, for example, a page fault or a general protection fault," you know, like if an instruction has memory that's been paged out. They say: "...requiring a roll-back. Already, without prediction mechanisms, processors sometimes have to flush the pipeline, for example, upon interrupts. With prediction mechanisms, there are more situations when partial pipeline flushes are necessary, namely on any misprediction. The pipeline flush reverts any architectural effects of instructions, ensuring functional correctness. Hence, the instructions are executed transiently: first they are, and then they vanish..." Then they say: "...(i.e., we call this transient execution).

"While the architectural effects and results of transient instructions are discarded, microarchitectural side effects remain beyond the transient execution." In other words,

that's the point, is that all of the focus has been on this, like, not looking at the man behind the curtain, only looking at the instructions being executed and making sure you're correct there. So correctness at this top level is what Intel has gotten right. But there was an ignoring of the consequences of the fact that the man behind the curtain that was doing all of this ended up with the right instructions having the proper effects. But the underlying mechanism is so complicated now that it wasn't being rolled back. There was debris left in what they called the "microarchitecture," not the macroarchitecture.

So they say: "This is the foundation of Spectre, Meltdown, and Foreshadow. These attacks exploit transient execution and encode secrets in the microarchitectural side effects, for example, in cache state, to transmit them to the architectural level to an attacker. The field of transient execution attacks emerged suddenly and grew rapidly, leading to a situation where people are not aware of all variants and their implications." And this is the formal way of saying what I said earlier, which is, you know, it's like, everyone went "Holy shit" and ran around and quickly patched some stuff to fix the obvious first things that people found. But it's until now, like today, that we have a full mature understanding of the broader picture, what really this is all about.

They said: "This is apparent from the confusing naming scheme that already led to an arguably wrong classification of at least one attack. Even more important, this confusion leads to misconceptions and wrong assumptions for defenses. Many defenses, for example, focus exclusively on hindering exploitation of a specific covert channel, instead of addressing the microarchitectural root cause of the leakage. Other defenses critically rely on state-of-the-art CPU features that have not yet been thoroughly evaluated from a transient security perspective. We also debunk implicit assumptions, including that AMD processors are immune to Meltdown-type effects, or that serializing instructions mitigate Spectre Variant 1 on any CPU.

"In this paper, we present a sound and extensible systematization of transient execution attacks - Spectre, Meltdown, Foreshadow, and related attacks. Using our universal decision tree, all known transient execution attacks were accurately classified through a universal and unambiguous naming scheme." And that's the picture at the top of this article that I put in there, and it's from their paper, and the link to the PDF is here if anyone wants more.

"The hierarchical and extensible nature of our classification methodology allows the easy identification of the residual attack surface, leading to seven new transient execution attacks - Spectre and Meltdown variants - that we describe in this work. These seven new attacks have been overlooked and not yet investigated so far. Two of the attacks are Meltdown-BR, exploiting a Meltdown-type effect on the x86 bound instruction on Intel and AMD; and Meltdown-PK, exploiting a Meltdown-type effect on memory protection keys on Intel. The other five attacks are previously overlooked mistraining strategies for Spectre-PHT and Spectre-BTB attacks.

"We demonstrate all seven attacks in practical proof-of-concept attacks on vulnerable code patterns, and evaluate them on processors of Intel, ARM, and AMD. We also provide a systematization of the state-of-the-art defenses. Based on this, we systematically evaluate defenses with practical experiments and theoretical arguments to show which work and which do not or cannot work. This systematic evaluation revealed that we can still mount transient execution attacks that are supposed to be mitigated by rolled out patches. Finally, we discuss how defenses can be designed to mitigate entire types of transient execution attacks."

And that's the beginning of 16 pages of all of the details. So anyway, here we are. It's been a year of this. And really it's not surprising because this is not easy stuff. If this was simple, it wouldn't have taken, what, 20 years for someone to finally say, you know, I

can see what's going on over in that other core. Is that what you meant or intended? And of course we know this 2018 has been the year of recognizing that it's possible to leak across processors, or across threads in a single processor.

Now, and again, we've also said, as I reiterated earlier in this podcast, that it's not clear to me that, as long as a browser cannot do this stuff, and there has been mitigation immediately added to our browsers, that's really the only place where the typical end user has - it is hosting code in their system in some fashion that they have really no control over. We're all downloading apps. And these days, certainly the listeners to this podcast are a little more skeptical than we were 13 years ago, before this podcast began dissecting all of this. I mean, I double-check where I'm downloading from. I try to find, if there's some download captain site for some Intel thing like a driver I need, I try to go get it from Intel. I don't want to get it from download captain.

So we have a lot of control over our own machines. And as I've said, this is the main concern are cloud providers hosting virtualized systems where multiple entities may be running code in the same platform. Which is why a lot of this attention has gone to breaking out of sandboxes and breaking out of virtual machine boundaries. I mean, that's how these things have been applied. So I don't think it's ever been the case that any of these problems affected end users. And Leo, as you and I have often reminded ourselves and anyone within earshot, there's never been an instance of any of this in the wild. No, I mean, unlike zero-day exploits that we're often talking about finding, like, ooh, crap, routers are being taken over by the hundreds of thousands, or Docker is falling. No, never. Not once has any of this actually turned into an attack that anyone found.

Leo: And it's because it's difficult that - tell me if I'm wrong, but my guess and understanding about this is that all these attacks, whether it's the hyperthreading attack that just came out, or Meltdown or Spectre, are timing attacks like Rowhammer where you kind of can deduce, I mean, this is crazy that this even works.

Steve: I know.

Leo: The information in a content stream, not because you can see into it, you're deducing it based on timing.

Steve: Yes. There was a stutter in the execution of your code that was of a different stutter duration.

Leo: That's, I mean, I don't blame - and honestly, I know Intel, you know, that this was theoretical before they even started doing speculative execution in 1991. Somebody wrote a paper.

Steve: Right.

Leo: But it seemed so farfetched that anybody could make this work. Now, the fact is, once somebody does make it work, if they can write a proof of concept, they can then be script-kiddie-ized.

Steve: Yes.

Leo: It's conceivable that it could get out, and somebody who is not in fact capable of figuring out what the content of a stream is from stutters would be able to use it. But so far nobody's done that. I don't know if that's because the researchers - and this is all, by the way, highly academic research. I think most of these guys, really they're just trying to get a PhD. It's not so...

Steve: I would give these guys 12 PhDs. Wow.

Leo: Yeah, I mean, it's brilliant. But maybe they haven't given enough details that you could weaponize it. I don't know why it hasn't leaked out. But it's incredibly hard to do, at least initially.

Steve: Yeah. My guess is this is sort of state actor level. This is, I mean, where you would put this would be if you could arrange to get your code running on a Google server and get a Google private key, that kind of thing.

Leo: Right.

Steve: Where the value is really high for something really well protected. And again, it's not somebody who wants to run coin mining somewhere.

Leo: Part of the reason for that, though, is it's just not worth the effort because there are so many easy ways to do it, to hack somebody's computer, that why go to all that effort. State actors have a reason to do it and the wherewithal to do it.

Steve: Right, yeah. And I do think, though, that this also, I mean, this is one of the core lessons the industry has learned, as was perfectly framed by Bruce Schneier, who said, famously, and we've often quoted this, "Attacks never get worse. They only ever get better." And so if this wasn't addressed, if the bad guys didn't know that it was being addressed, then they could roll up their sleeves, and there would have been an opening to weaponize this. And so I'm glad this has gotten the attention that it has.

Leo: Yes, absolutely.

Steve: Okay. With all that said, we have the problem, and it bugs me, I mean, I've got my little InSpectre tool that is woefully now out of date. People complain that I haven't gone back and kept it current. But everyone also wants new SpinRite, and so I've got to get SQRL finished first. But it bugs me that I've got machines which Intel will never update, I mean, not Intel. The manufacturer will never update them. They're still useful. They're still workable. They're still fine. But I don't have updated BIOSes for them. So I can't ever get all happy green on InSpectre. It's like, sorry, you don't have the firmware.

There's a solution. There is something open source available called the Intel Microcode Bootloader. This is not something for non-techies. This is maybe, I would imagine, this will be of interest to our listeners, which is why I'm bringing it up. But it could be of

serious interest to anybody who's running older hardware in a shared VM hosting cloud that is concerned about these mitigations, and they are not available.

Okay. So what this is, first of all, the reason a BIOS update can fix the microcode is that all Intel processors have firmware which can be loaded into them when they are powered up. So they come with a ROM, but the ROM is slower than the internal RAM. So the ROM is copied into the RAM for maximum execution speed. That contents of the RAM can be patched or changed. And so when a BIOS powers up, it contains patches for the processor that it knows is on its motherboard, and it applies those patches on the fly. In this case, all of these updated BIOSes contain updated microcode that is being reloaded into the processor's firmware RAM every time it powers up. So the problem is it has to be an "every time you power up" solution. The BIOS can do that. But so could something else that boots first and then transfers control to another bootable thing.

Leo: I'm sure this is trustworthy, but what could possibly go wrong?

Steve: Okay. So a developer has created a preboot patch for Intel microcode providing the latest Intel processor microcode for, get this, 392 Intel CPUs produced from 1996 to 2018. So way back. This Intel microcode bootloader provides a workaround for the "my BIOS hasn't been updated for Spectre and Meltdown and probably never will be" problem. It dynamically updates the microcode every time the system is powered up. This Intel Microcode Bootloader is based on - and this is news - based on Intel BIOS Implementation Test Suite, BITS, B-I-T-S, BIOS Implementation Test Suite. Put that in your Google, Leo. So that users no longer need to modify BIOS UEFI ROMs to stay protected from security vulnerabilities, bugs, and errata.

So I thought, okay, wait. What is BITS? So I went looking, and I found an Intel link. Get this. The Intel BIOS Implementation Test Suite provides a bootable pre-OS - this is Intel. I'm reading from Intel. This is Intel's page. The Intel BIOS Implementation Test Suite provides a bootable pre-OS environment for testing BIOS and in particular the initialization of Intel processors, hardware, and technologies. BITS can verify your BIOS against many Intel recommendations. In addition, BITS includes Intel's official reference code as provided to BIOS manufacturers, which you can use to override your BIOS's hardware initialization with a known good configuration, and then boot an OS.

So they say, under who should use BITS, they said: "You might want to use BITS if you're a system or BIOS developer, and you want to validate that your system meets Intel's recommendations; you're an OS or application developer building on technologies provided by Intel platforms, and you want to check if your system, or one of your user's systems, has configured those technologies correctly; you're an advanced user or developer, and you want to check your BIOS to see if it configures Intel hardware correctly and, if not, to make a stronger case to your BIOS vendor to get it fixed; or you need to poke hardware in a low-level way, and you need a pre-OS environment to work in to avoid OS interference." Anyway, I'm just stunned that this thing exists, and it's available: biosbits.org, B-I-O-S-B-I-T-S dot org.

Leo: There's apparently something similar for AMD.

Steve: Cool.

Leo: Does the same thing, yeah.

Steve: Cool. So they said: "BITS consists of a modified GRUB2 bootloader, with many additional commands to probe and manipulate hardware configuration, as well as scripts using these commands to test and reconfigure hardware. BITS supports scripting via Python and includes Python APIs to access various low-level functionality of the hardware platform" - oh, can you get in trouble now - "including ACPI, CPU and chipset registers, PCI and PCI Express. You can write scripts to explore and test platform functionality using the full power of Python in 32-bit ring 0, without any OS in the way, and without recompiling BITS or writing custom C code. See our Python scripting guide for more information." Anyway, I'm just...

Leo: Oh, boy. Oh, boy.

Steve: So on top of this real foundation, there is now an Intel Microcode Bootloader. The guy has on the page that I've got a link to, and I saw you brought it up during the podcast, instructions: "Format a USB flash drive with a FAT32 file system. Extract the archive to the USB flash drive and run install.exe to make it bootable. Enter the BIOS/UEFI, assign the USB flash drive as the first boot device, and enable legacy boot mode. The bootloader will regularly update the microcode and load the OS."

So the idea would be, if you were an end user, or if you were somebody with a cloud system, you would get a little thumb drive, set this up, stick it into a USB slot, and forget it. Or actually, there are motherboards now with internal USB, so you could just stick the little thumb drive on the motherboard on one of the internal USB sockets and just arrange to have it boot first and then let it go ahead and continue to boot your OS as it usually would. That thing will update your microcode where your motherboard wouldn't, and then execute your OS afterwards. So anyway, I just wanted to put it on everybody's radar. I thought it was very cool. And this BITS thing is making me think about, ooh, SpinRite, hmm.

Leo: Hmm. Oh, yeah, you're right. Now you're going to write SpinRite in Python. Hmm.

Steve: Hmm. So Firefox Monitor is a service that we've never really talked about. It's not been on our radar. But a forthcoming feature in Firefox, if you'll pardon the pun, popped up, which is available in multiple languages, and Mozilla's excited about it. Last Wednesday they announced the addition of a new feature for Firefox. The first time and only the first time a Firefox visitor goes to a site that has suffered from a publicly reported data breach within the past 12 months, a pop-up notification will appear notifying the user of a prior data breach at that site. And this is like - and I have a picture of a sample from their announcement in the show notes where they just use example.com, and it shows a little popup, and it asks the question, have an account on this site? More than - and then there's a number that would be filled in based on the real data, here more than 500,000 accounts from example domain were compromised in 2018. Check Firefox Monitor to see if yours is at risk.

And so there is a - you can click on Check Firefox Monitor, or you can dismiss it, or you also have the option to turn off this breach notification system for all future. So the user can take note of and dismiss the notification, or it can elect to immediately jump over to the Firefox Monitor site, where they can confidentially enter their email address and have Troy Hunt's excellent HaveIBeenPwned site as a service check for any exposures of the user's email. And anyone not wishing to receive these alerts on any site can simply choose "Never show Firefox Monitor alerts" by choosing, as I mentioned, that dropdown

arrow on the notification. Mozilla has said that this functionality will be gradually rolled out to Firefox users over coming weeks.

And we've not talked about Firefox Monitor before. That's at monitor.firefox.com. And it looks like it offers its own front end to Troy's, you know, the HaveIBeenPwned facility, and Troy publishes a web API to facilitate such third-party access, making it a service. At monitor.firefox.com, users can sign up with Firefox Monitor using their email address to receive proactive notification if a breach occurs which does involve their email address. Which that's kind of cool. Troy does not himself offer that service. But if you are a Firefox user, or I guess even if you're not, you can go to monitor.firefox.com, give them any email addresses you want them to watch. And if breaches occur, they'll proactively check to see whether your email address is among them and send you a note, if so. Which is very cool.

So a couple weeks ago we talked about the unpatched Docker flaws which would potentially open the door to anybody who found them. And not surprisingly, Juniper Networks Threat Labs recently discovered and reported on malware in the wild which is searching for exactly that: misconfigured, publicly exposed Docker services on the Internet, which it is infecting with Monero miners. I won't get into any super detail. I'll note that Docker by default is not on the Internet. It is only using Unix sockets, which is to say using the sockets interface in Unix to do within single machine communications, not on the Internet. So it's necessary for someone to explicitly bind the Docker APIs to Internet sockets in order to make them available. If done, then those are 2375 and 2376, which if enabled provide both unencrypted and unauthenticated access to the Docker APIs.

So the point is that - and maybe people doing this don't understand that, because the presumption was that this API would never be publicly exposed, it is not authenticated by default. The presumption is, if you're running on the same machine, you inherently have access to the APIs of that machine. But when bound to Internet ports, and if it's on the WAN interface, facing the public, everybody can get them. And it looks like many people have made that mistake.

Once a new host is infected, it starts looking for other accessible hosts, both on the public Internet and on any internal networks the host has access to. So that of course will typically be a corporate Intranet, and the infected victim machine ends up servicing as an unwitting bridge from between the public Internet and the internal private Intranet. Juniper has gone through and decrypted all of the scripting that is going on. They described the infection chain as largely living off the land because it leverages well-known system-provided utilities to spread its infection and carry out all its activities. They wrote that some of the system utilities that it uses are Docker, Wget, cURL, Bash, iproute2, Masscan, apt-get, yum, up2date, pacman, dpkg-query, systemd, and so forth, all which are familiar names to people who use Unix or Linux machines.

Juniper has dissected and described the operation of the malware in detail and noted that it downloads a stock version of the Monero miner bash script. It runs MoneroOcean's Monero miner bash script, which is hosted on Pastebin, and executes it. And then once that's done, it begins scanning all available networks for any port 2375 and 2376. When it finds any, it dumps it to a local.txt file, and that contains the list of IP addresses which it then proceeds to infect. So anyway, again, what we're seeing is, if something can be infected on the public Internet like this, it's not making people lots of money, but it's free after the set this up and turn it loose. So that's what happens.

I mentioned at the top of the show that we had recently ratified HTTP/2. For years we've been at HTTP/1.1. That's often the protocol which is recognized when a server responds to an HTTP query. It'll send back an okay, 200 okay, and also HTTP/1.1. Now HTTP/2 is where we are. And as a consequence of some recent communications among the IETF,

the decision has been made and consensus reached about HTTP/3. And it will use, sort of dramatically, actually, UDP, not TCP, as its underlying...

Leo: Wait a minute. What?

Steve: Yes. Not TCP as its underlying transport. Mark Nottingham, who is the chair for both the HTTP working group and the QUIC (Q-U-I-C) working group for the IETF proposed renaming what is now known as HTTP-over-QUIC (Q-U-I-C) to HTTP/3, and the proposal appears to have been broadly accepted. HTTP/3 will have QUIC as an essential, integral feature such that HTTP/3 will always use QUIC (Q-U-I-C) as its network protocol. Okay. So let's back up here a bit.

Leo: Ooh.

Steve: Yeah.

Leo: That's a big change.

Steve: That's a huge, huge change.

Leo: The reason I thought they used TCP is for reliability; right? Because you have a SYN and ACK packet. UDP is used for streaming because you don't have to have the ACK. You just keep streaming.

Steve: Right. It was reliability in the sense of remember that the [crosstalk]...

Leo: Not caring if you got it, basically.

Steve: Right, right. Exactly. Like you and I have this conversation going over UDP because it is lighter weight. And if a packet doesn't arrive, it only is a tiny bit of sound worth of packet that was lost, and the codec at the receiving end is smart enough to just sort of repeat what it already had and fill the missing time. And if the loss is bigger, then people will hear kind of a twang, like we've often in the older days heard from cell phones. But in general, the idea being that UDP is unreliable. Well, you could build reliability on top of UDP if you're smart. Okay. So let's back up a bit.

As we know, traditional secure connections over the Internet start with a DNS lookup to map the domain name to the Internet address. Originally that was IPv4, and increasingly it's IPv6. But address lookup is typically cached, and it's not really about connections. So that's out of scope for this.

I have a picture here in the show notes of how a secure TLS connection is currently set up on the Internet. Once we have an IP address, multiple packets are sent back and forth to first establish a TCP connection. Sequence numbers are exchanged, and IP level features are negotiated between the end points. And as we know, there's the famous TCP SYN, where the client, the connection initiator, sends its sequence number, that is, a random 32-bit number from which it will number all bytes that it then subsequently

sends. The server responds with an ACK for the client's SYN, and its own SYN in one combined packet with two of those bit flags sent in the TCP header. So that's called the SYN ACK packet. And so upon the client receiving it, they each have each other's sequence number. Then the client responds with its ACK to acknowledge the server's SYN, and at that point the TCP connection is said to be open.

The client then typically like in this case in a web browser scenario wants to establish - it wants to go from a TCP connection to a TLS connection. It wants to establish security. So it sends a client hello packet, which contains a cryptographic nonce which it has chosen, along with a list of all the cipher suites for the version of TLS that it supports. The server receives the client hello. It looks through its prioritized list of cipher suites in the order it would like to use them for the first one which is in the client's list. So it chooses one that it likes and supports. It also chooses its cryptographic material, its nonce, and returns that in a packet with multiple flags.

A server hello, the certificate - oh, I forgot. The client's hello also specifies in the SNI, the Server Name Indicator extension to the client hello packet, the name of the service it wants. That allows a multiply home server to rummage around in its certificates and find the one that the client wants and to return the certificate to assert its identity to the client. So it sends a certificate, the server hello, back to the client, which contains its crypto information.

So now they've both exchanged crypto information. They've agreed upon a cryptographic protocol. The client needs to respond with an acknowledgement of the cipher spec that the server chose with a so-called "change cipher spec" message to say, okay, I accept what you've returned. I am changing over to that cipher. The server needs to do the same thing. It needs to acknowledge the client's message and say everything from now on is under the new cipher that we have agreed on.

So they both bring their ciphers up using the nonces that they've exchanged, which allowed them to negotiate a secure secret key under which they then encrypt all data over not only - so the first layer's TCP. The second layer is TLS. Now the TLS tunnel is up, and they can exchange application data with authentication and secrecy. Whew. QUIC, Q-U-I-C.

Leo: Such a short acronym for so much stuff.

Steve: Well, QUIC does that in one packet.

Leo: Well, that's not QUIC. That's [crosstalk] old school. That's the old way of doing it.

Steve: So believe it or not, QUIC, which stands for Quick UDP Internet Connections, Q-U-I-C, is able under the proper conditions and preconditions to achieve the same with a single packet from the client to the server, everything bundled up into one packet. So for this we have Google to thank. Recall that Google's earlier SPDY, S-P-D-Y, technology also proved itself worthy and formed the basis of HTTP/2, which has now become an IETF standard, that is, in the process of getting fully supported. HTTP-over-QUIC, which is now - now the IETF has said that will be HTTP/3, is a rewrite of the HTTP protocol that uses Google's QUIC instead of TCP as its base.

QUIC is Google's complete rewrite and rethink of that entire protocol stack which I just went through which combines everything - IP, TCP, UDP, TLS, and HTTP/2 - into a single

amalgam. So Google has proposed that QUIC might eventually replace both TCP and UDP as the new protocol of choice for moving binary data across the Internet. Tests have demonstrated that QUIC is both faster and more secure because it also is an encrypted by default system, that is, there is no nonencrypted version. And it uses the just recently ratified TLS v1.3 protocol built in. So it was proposed as a draft standard at the IETF in 2015. The version of HTTP-over-QUIC, which was a rewrite of HTTP on top of QUIC instead of on top of TCP, was proposed a year later, in July of 2016.

So these things take time. Here we are, nearly 2.5 years later, and it now looks like it's going to become the next big standard when we're ready to move away from HTTP/2. The support for HTTP-over-QUIC, that is, this next generation, was added to Chrome 29 and Opera 16. It's also supported in the LiteSpeed web servers, and Facebook has started adopting the technology. So it looks like it's going to happen. And essentially, as we know, anyone who's looked at the developer console of their browser and brought up any contemporary web page knows that stuff is coming from all over hell and gone. I mean, it is just, you know, today's web pages are nuts with where they're pulling all of their assets. Every one of those requires a negotiation with a different domain and remote server, bringing up TCP, bringing up TLS, and then finally being able to make the query in order to get going.

What this does, again, given some cached agreements among endpoints, and QUIC manages all of that, it allows a single packet to be sent to a service in order to request an asset and have that asset returned. So we're talking about once we get this, once both ends have made the move over, significantly quicker loading web pages. Which of course is why Google was behind this from the beginning, as they would like everything to be faster. And who wouldn't?

Last week, as I mentioned at the top of the show also, we talked about the cool new app from Cloudflare for iOS and Android devices, and https://1.1.1.1, or one dot one dot one dot one. And I stumbled over my recognition during the podcast that it was a secure connection. And it was like, what? So my first erroneous assumption was that, well, if it's a secure connection, it has to have a certificate. And if it has a certificate, then dot one...

Leo: It's a domain, yeah.

Steve: It has to be a domain name. And of course that was wrong because we all - how many times have we put http://192.168.0.1 or .1.1 or something into our browser in order to access our router. As we know, dotted numbers, dotted quad numbers are IP addresses. They're not domain names, and they're not legal for domain names. So "1" cannot be a domain name. It has to be part of a dotted quad.

Leo: This is John Graham-Cumming's email. He probably sent it to you, too.

Steve: Yeah.

Leo: Thanks for viewing. Happy to tell you more. 1.1.1.1 in the IP address of the site; and, yes, we have an SSL certificate for an IP address. That's the key; right? You can get one for an address.

Steve: Well, yes. And the secret is...

Leo: They have friends.

Steve: Well, yeah. They have friends. And it's the SAN field. We've talked about this extension to certificates often. That's the server alternative name. And, for example, like GRC's certificate has GRC.com and I don't know what, www.grc.com. I don't think it has star. It might have www.grctech.com. Anyway, the point is multiple alternative names for the same service. And it turns out, there IP addresses are legal. So you can put an IP address in the SAN field. Let's Encrypt will not allow that. And I had forgotten, if I ever knew it, that it was possible to put an IP address in there, but it is.

So the answer to this mystery is that when we went to look at it, that wasn't the name of the certificate. The name was *.cloudflare or whatever it was dot something or other dot com. And this was a DigiCert certificate. And they arranged to have 1.1.1.1 placed in the enumeration of alternative names in that certificate. Thus they're able to offer a TLS connection for their stuff. So very cool. And mystery solved. And I wanted to close the loop on that.

Christian Alexandrov on the 19th tweeted: "@SGgrc One short SpinRite testimonial." He said: "If an HDD [hard disk drive] wants to die, it will die. My regular use of SpinRite on a drive gave me a lot of warning. So I had the time to save all my data which I care about. One more lesson for people: Regular use of SpinRite warns you when something is going to happen."

And anyway, Christian, thank you for that. And I will reiterate that we know that drives have that SMART data, the self-monitoring analysis and reporting tool, S-M-A-R-T. The problem is it's only meaningful when the drive is being asked to do something. If you look at the SMART data while the drive is sort of idling, maybe there's some information there, but it doesn't really tell you what's going to happen when you ask the drive to do some work. If you make it sweat, that's when things happen. And it's years of having that in, like, what, now, 14 years of that being available because I added that in SpinRite 6, has demonstrated that, when you're running SpinRite, if the drive is in trouble, you will start seeing red in the health bars that SMART is reporting.

Essentially, if running SpinRite pushes the drive's health down, that gives, I mean, the drive is still running. Everything is still fine. But a healthy drive won't have its health pushed down just by asking it to do what it was born to do. So it really does provide a sort of an analog sense for how much left there is in this drive. And as Christian says, if you use SpinRite and take a look at that SMART data, after the drive's been running for a while under SpinRite, if everything is still happy in the green, you're okay. At least the drive is not saying, boy, you know, I'm having to work here in order to just read some data, which was what I'm supposed to be able to do easily. That's, as Christian says, a nice early warning system for telling you maybe it's time to make sure you have a current backup.

Terry Daniel tweeted: "@SGgrc Since SQRL tokens are linked to website name, is there a solution in SQRL for the problem that occurs when a web property changes its name," he says, for example, "United to Continental, or Ofoto to Kodak Gallery." It's a great question, Terry. And it's one that we've talked about in the SQRL forum and have a solution for. It is the case that all of the security that SQRL offers, which as far as we know is absolute, that is, I haven't talked about it except in passing, we'll get around to it here in the future once I'm able to stop teasing people with this and actually let people play with it or have it more widely available to play with.

As far as we know, it is unspoofable. I mean, really, we solved that problem. And but that requires a tight binding between your identity and the domain name, or that absolute unspoofability guarantee would be broken. But there is the ability to have an

entity who has had to move itself to a different domain to bring SQRL users from the old domain to the new domain. So the idea would be, if United purchased Continental, what would happen would be if a SQRL user then logged into United, United server would notice that, oh, someone's trying to log in with SQRL, but we don't recognize them. The result of that would be a page that says, "Hi there. You're trying to log in here with SQRL, but you might be a Continental user. Try again." And so that second page would be the SQRL at the Continental domain which United purchased.

And so the idea is that one last authentication at the domain that's being left, that's being abandoned, that is being moved from, allows United to identify the user under both the United.com and Continental.com domains because, after all, the person just tried to use SQRL at United, so United has their SQRL identity there. That allows them to know who they were under the Continental domain and essentially migrate them without the user having to do anything except just try to log in under the old domain, and it's able to happen automatically. Or, if they go to Continental.com, which is still up, then it's able to move them over. So great question, Terry. And SQRL does have a solution for that, which is a little bit of a double step, but only has to happen once, and then you're known under the new domain.

John Baxter says: "Hi, Steve. Do you have any information on how long external SSD devices can sustain their contents while not plugged in to power? Specifically, the Samsung T5. Clearly they aren't permanent, as the capacitors have to be leaking charge at some rate. Two weeks seems to be okay. I haven't found information at Samsung or in reviews." He says: "At two weeks, I could be running through spares way too fast, but I think I would have seen complaints. One solution might be to store the devices plugged in on power, but at some point the better choice would seem to be spinning drives."

Okay. So there are a couple misconceptions here. I've referred to SSDs as having electrons stranded on a little pad, essentially. That's different from a capacitor. DRAM is a capacitor which is leaking charge very quickly. So we know that it's surprisingly slow so that, for example, if you spray it with Freon, you can reduce the leakage of the DRAM's capacitors long enough to move it into a different piece of hardware sometimes and then bring it back to life and not have lost a lot of the DRAM's data. SSDs are not capacitors. They are an isolated gate technology where there is some leakage from electron tunneling, but it is really, really low. And we've talked in the past about how, for example, there is a temperature sensitivity. The hotter they are, the faster they leak. But it is in years, not in weeks.

Leo: Camera flash is not getting refreshed, and it sits for hours, days, weeks, months, and years. And, I mean, as far as I can tell, if I took a picture and leave it on an SD card, it's there a year or two later.

Steve: Yes.

Leo: So it's the same, basically, the same kind of technology; right?

Steve: Yes, it is the same technology. And for what it's worth, Leo, if your refrigerator has any space available...

Leo: I would, but it's filled with, I don't know, what is it, Palm Pilots?

Steve: Palm Pilots, yes.

Leo: Yeah.

Steve: So anyway, so there is a temperature-sensitive leakage. So actually not having them plugged in would be better than having them plugged in.

Leo: Oh, interesting. [Crosstalk].

Steve: The colder, yes, the colder they are, the less quickly the stranded charges would tend to bleed off. But it is in years. And what you really need to do is like every year run SpinRite on them.

Leo: Oh, good idea.

Steve: Because it is only by reading and rewriting an SSD that its charges are refilled.

Leo: Ah, interesting.

Steve: That's the thing that recharges the SSD's storage over time. And what's happened, of course, is that as we've pushed density higher, those SSD cells that used to be SLC, Single Level Cell? Now they are MLC or TLC. MLC was four levels of charge, and TLC is eight levels of charge. So each bit cell now stores three bits. Which means that it's more important that you distinguish the exact amount of charge in the cell than ever before.

That is, unfortunately, a tradeoff has been made for density over reliability, which is, for example, why I can't get them anymore, but the SSDs that the GRC server is running on are SLC SSDs. And even then they're in RAID 6 because you know I'm belt-and-suspenders and SpinRite in order to have the greatest level of reliability. But anyway, not to worry, and do keep them cool. If you really are worried, stick them in the fridge. It won't hurt them, and it will reduce the rate at which the charge is leaking from an offline SSD.

James tweeted: "Listening to this week's episode of Security Now!. Could you produce an app-specific public key that could be given to the bots" - oh, he's referring to our discussion about having a bot use SQRL if it needed to log into a site on your behalf - "could be given to bots that use your SQRL identity and have the main protocol authorize the use of that key for a specific app when you're connected to an authenticated session. You could then manage on a per-site basis the apps that have authority to use your identity on that site, much the same way we use app-specific passwords on Google or other services. You could then prune your app-specific keys in the SQRL app with any revocations being communicated to the site via the protocol and signed with your master identity to authorize the revocation." Whew. Okay.

So what he's saying essentially is could you authorize a different SQRL identity, which would be the bots, to log in on your behalf at a site? And the answer is something we called "managed shared access." And it is what I brought online Saturday, three days ago. That's this thing that I've been working on for the last 90 days. We had a definition

of it in mind. We had never created an implementation. That's what I wrote. I wrote a whole new server and an API, essentially a design to support that.

The problem is right now in the world where identity is, I was going to say loosely bound to people, but really not bound to people at all because it's not even a retina or a fingerprint or your face. It's just your username and password. People commonly share their usernames and passwords with other, for example, family members, when they need to share, they want to grant access to a website like Netflix, for example, or Mom and Dad both know the username and password for the bank. Well, in a world which we are arguably moving toward, where identity is more tightly bound because we've got biometrics of one form or another, or something like SQRL, which is a proxy for you, that's inherently hostile to sharing your identity. You don't want to share your SQRL identity with anyone because it doesn't just represent you at one site. It represents you everywhere. That's the benefit and the power.

But with that comes this problem that we're used to just giving away usernames and passwords when we want to share. So what we need is we need services that support SQRL to have as an option, not necessarily, but as an option, the ability to allow multiple SQRL identities to all share an account. And we have that. It's a well-defined spec. Everybody who's listening to this will be able to play with that. The guys in the SQRL newsgroup since Saturday have been playing with it. And it's very cool. It is no decrease in security, and it provides a means for sort of an owner of the account to invite others to use their SQRL identities there and to curate who has what level of access and to eliminate people after they've wandered off or you no longer want to share your access.

So managed shared access. It's in the API. It exists. And James, you foresaw, essentially, a need that we did, too. And it's part of the solution. And as I've said before, and I'll be talking about it again in a minute when we talk about what Troy Hunt was saying, I believe every problem has been solved. Other solutions have not solved every problem. I think every problem has been solved. And it's why I think it has a chance to succeed. We'll see. I'm happy to let the world judge that.

Yodar44 said: "Steve, in SN-689 you focused on BitLocker on SSDs." That of course was last week's podcast, "Self-Decrypting SSDs." He says: "Do the same concerns apply to BitLocker on spinning drives?" To which I would answer the same concerns, absolutely. What we don't have is research for whether spinning drives have been as negligent, frankly, as at least we know Samsung, and was it Corsair, the two brands of SSDs that those guys dissected. It's not until you look, unfortunately, that we know what's going on underneath the covers. Because these guys looked, we realized, oh, crap, this is not providing actual security, which is one of the reasons why it's a problem that these things are black boxes to us. These guys went to great lengths to look inside. So I would say yes, until we know definitively that a drive has implemented its encryption properly, you cannot trust a drive. You have to revert to software solutions. And CPH...

Leo: Which drives, just to recap, do it right?

Steve: Well, we don't know of any that do it right.

Leo: Oh. There you go.

Steve: Yeah. But we're assuming that Samsung, and was it Corsair? Doesn't feel like it was Corsair. Corsairs are...

Leo: Crucial?

Steve: Crucial, yes. Samsung and Crucial have updated - actually I think I saw that the MX300, that's the Crucial drive, had not yet been updated. But when they update their firmware, we hope they do so in a way that would satisfy these academics who looked into them and found them wanting. But, I mean, what we really need is - because this is not, like, implementing this correctly, that's not proprietary. It's not something that the drive needs to keep secret from everybody else. We ought to have, and the Trusted Platform Computing group ought to provide, an open source reference for here's how you implement this overly complex security standard which we defined. So just code this into firmware and do it.

And if we knew that that had been done, then good. It'd be like the reason we trusted TrueCrypt and we trust VeraCrypt is it's open source. And it's been vetted multiple times. So we know what it does. Drives, we've been assuming, oh, they say it's hardware, so it must be better. No. Now we know that, you know, that research put the lie to that assumption.

Leo: Yeah. The research was Samsung and Crucial didn't do it right, and those companies responded by patches. But I'm reading Samsung's support page. It says for non-portable SSDs like the Samsung EVO drives that are very widely used, in fact it's my number one choice...

Steve: Yes, yes.

Leo: ...we recommend installing encryption software. So they did update. That's the portable SSDs like the T5 our previous guy was talking about, and the T3. And I use those. But you first have to reinstall the portable SSD activation software and then update the firmware. And if you have a T1, well, you have to contact the Samsung Service Center. So the portable SSDs are, with maybe some effort, made secure. But it doesn't look like they have any plans for making the most common SSDs in use, the EVOs, secure.

Steve: Wow. Wow.

Leo: So you should use, I mean, the answer is you should use VeraCrypt.

Steve: Yes, that is the answer.

Leo: That's what Samsung says.

Steve: Yes. Or BitLocker, when you tell it not to use hardware. Which...

Leo: Oh, yeah, you can tell BitLocker not to do that, yeah, okay, yeah.

Steve: Yes. CPH said: "Steve, Windows 7 BitLocker is unaffected by the hardware encryption problem because Windows 7 BitLocker doesn't support leveraging hardware encryption in the first place, so VeraCrypt is not the only recourse there."

Leo: [Crosstalk] Microsoft updates; right? Because now they know there's a flaw, and they should just update all BitLocker implementations to do software encoding.

Steve: Yes, they ought to just back off of, yes, that's exactly right. I think that makes a lot more sense. I'm going to skip one because it's overly long and not necessary. Mikael Falkvidd said: "Re SSD encryption," he says, "the key cannot be derived from the password." He says: "The password must be used to unlock the key. Otherwise it would be impossible to ever change the password."

Leo: Good point.

Steve: "And it would be impossible to start using a password without wiping the entire drive." He says: "I might have misunderstood your description," so he says, "/rant," meaning closing rant. He says: "If that's the case, I'm sorry, but I would love a clarification." So the clarification is there's an additional what we would call a "level of indirection." So the key that you are using is not the one that encrypts the drive. That one is the drive's master key, which never changes.

So the way you change your password is you give it the old password, which is used to decrypt the key, not unlock it, to decrypt it. And now it's holding its breath, and it's decrypted. Now you give it the new password, and that decrypted key is then reencrypted under the new password. So that's how you perform a change of password without losing the contents of the drive is you have both, you know, you always want the old password to prove you're you, and then the new password. So that's a similar sort of a straddle, kind of like what SQRL does from one domain to another, is you decrypt under the old password and then immediately reencrypt under the new password. And that way you're able to make a change.

Tinzien says: "Hi, Steve. In the past you advocated using Windows Defender plus Malwarebytes. With the new Win Defender sandbox, are you suggesting no Malwarebytes is needed/is a vector for problems? Or is this an omission? Thank you for your continued great work." And when I have suggested Malwarebytes it's only to use it transiently as a scanner, never to install it. So that's the source of the confusion is I do like Malwarebytes, and I have used it sometimes to scan a system just as an additional check when something seemed a little weird, just to make sure nothing was getting away from Windows. So I use it only to scan, never to install it permanently. And I think, as we've been saying now, you and I, Leo, for a while, I think Windows Defender is the answer for this.

Leo: Now for the fun final - what are you laughing at?

Steve: I'm just smiling.

Leo: Are you laughing at Mark Zuckerberg, who is enjoying the show today?

Steve: So Troy Hunt on passwords.

Leo: Oh, this is good. I read this article. I thought, Steve's going to have something to say about this.

Steve: Yeah. So just to recap, before I share what Troy said, three days ago, Saturday afternoon, I put what I've been working on for the past three months online, to be pounded on and commented upon by our wonderful group of testers, techies, and developers over in GRC's SQRL newsgroup. So far, since then, I've fixed a few bugs, and I have a short to-do list of features to add or tweak. This was an important piece of work since it defined and has now proven and verified that I got it right a minimal and workable generic SQRL service provider API which allows a web server to query an external SQRL provider for all of the site's SQRL support, so it creates a well-defined boundary. The provider still needs to be present on the web server's domain, but this allows for a clean externalization and a well-defined boundary between SQRL's authentication functions and an existing website.

Fifteen days ago, on November 5th, while I was working on this, so I didn't respond to it until I've come up for air, Troy Hunt, who as we know is the HaveIBeenPwned guy, and he's a prolific writer about security, and he knows what he's talking about, posted an article which he titled "Here's Why," and then he says "[Insert Thing Here] Is Not a Password Killer." And I have a link in case anyone's interested. But I'm sure if you put "here's why insert thing here is not a password killer" into Google, it'll find it. So this is what Troy wrote.

"These days I get a lot of messages from people on security-related things. Often it's related to data breaches or sloppy behavior on behalf of some online service playing fast and loose with HTTPS or passwords or some other easily observable security posture. But on a fairly regular basis I get an email from someone which effectively boils down to this: 'Hey, have you seen [insert thing here]? It's totally going to kill passwords!'" And Troy says: "No, it's not. And to save myself from repeating the same message over and over again, I want to articulate precisely why passwords have a lot of life left in them yet. But firstly, let me provide a high-level overview of the sort of product I'm talking about, and I'll begin with recognizing the problem it's trying to solve: People suck at passwords.

"I know. Massive shock; right? They suck at making genuinely strong ones, they suck at making unique ones, and they suck at handling them in a secure fashion. This leads to everything from simple account takeover (someone else now controls their eBay or their Spotify or whatever), to financial damages (goods or services bought or sold under their identity), to full data breaches (many of these occur due to admins reusing credentials). There is no escaping the fact that passwords remain high-risk security propositions for the vast majority of people. Part of the solution to this is to give people the controls to do password-based authentication better, for example by using a password manager and enabling two-factor authentication. But others believe that passwords themselves have to go completely to solve the problem, which brings us to proposed alternatives."

He says: "I don't want to single out any one product out there because the piece I'm writing is bigger than that, so let's talk about patterns instead. I'm referring to passwordless solutions that involves things like QR codes, pictorial representations, third-party mobile apps, dedicated hardware devices, or 'magic' links sent via email. I'm sure there are others; but for the purposes of this post, any pattern that doesn't involve entering a username and password into a couple of input fields is in scope. To their credit, some of these solutions are genuinely very good, technically very good. But what proponents of them seem to regularly miss is that 'technically' isn't enough. Despite their respective merits, every one of these solutions has a massive shortcoming that severely

limits their viability, and it's something they simply can't compete with. Despite its many flaws, the one thing that the humble password has going for it over technically superior alternatives is that everyone understands how to use it. Everyone."

Leo: Yeah. Not how to use it securely or well.

Steve: Yes.

Leo: But they do know how to use it.

Steve: Yes. It is literally the definition of the lowest common denominator.

Leo: Yeah, yeah.

Steve: And he says: "This is where we need to recognize that decisions around things like auth schemes go well beyond technology merits alone. Arguably, the same could be said about any security control. And I've made the point many times," he writes, "before that these things need to be looked at from a very balanced viewpoint. There are merits, and there are deficiencies. And unless you can recognize both, regardless of how much you agree with them, it's going to be hard to arrive at the best outcome."

He says: "Let me put this in perspective. Assume you're tasked with building a new system which has a requirement for registration and, subsequently, authentication. You go to the marketing manager and say, 'Hey, there's this great product called'" - and he has, you know, "[insert thing here]" - "'that replaces passwords, and all you have to do to sign in is...' And you've already lost the argument because the foremost thing on the marketing manager's mind is reducing friction.

"Their number one priority is to get people signing up to the service and using it because, ultimately, that's what generates revenue or increases brand awareness or customer loyalty or achieves whatever the objective was for creating the service in the first place. As soon as you ask people to start doing something they're not familiar with, the risk of them simply not going through with it amplifies and defeats the whole point of having the service in the first place."

And he goes on, and it's long, and I don't want to skip anything, but everyone should have the point of what he's talking about. I mean, I just saw him saying right here: "What I often find when I have these discussions is a myopic focus on technical merits. I'll give you an example from earlier last year where someone reached out and espoused the virtues of the solution they'd built." And I should mention I've never spoken to Troy, so it wasn't me. Because SQRL's not out yet.

He says: "They were emphatic that passwords were no longer required due to the merits of [insert thing here] and were frustrated that the companies they were approaching weren't entertaining the idea of using their product. I replied and explained pretty much what's outlined above. The conversation is going to get shut down as soon as you start asking companies to impose friction on their users. But try as I might, they simply couldn't get the message. 'What barrier? There's no barrier.' They went on to say that companies not willing to embrace products like this and educate their users about alternative auth schemes are the real problem, and that they should adjust their behavior accordingly."

Troy says: "I countered with what remains a point that's very hard to argue against: 'If your product is so awesome, have you stopped to consider why no one is using it?' Now, in fairness, it may not be precisely 'no one.' But in this case and so many other of the [insert things here], I'd never seen them in use before, and I do tend to get around the Internet a bit. Maybe they're used in very niche corners of the web. The point is that none of these products are exactly taking the industry by storm, and there's a very simple reason for that: There's a fundamental usability problem. This particular discussion ended when they replied with this: 'I think it is only negativity that doesn't allow positiveness to excel.'" He says: "Ugh." He says: "I'm negative about stuff that's no good, yes." He says: "I dropped out of the discussion at that point."

Anyway, he says: "This is why passwords aren't going anywhere in the foreseeable future and why [insert thing here] isn't going to kill them. No amount of focusing on how bad passwords are or how many accounts have been breached or what it costs when people can't access their accounts is going to change that. Nor will the technical prowess of [insert thing here] change the discussion because it simply can't compete with passwords on that one metric organizations are so focused on: usability. Sure, there'll be edge cases, and certainly there remain scenarios where higher friction can be justified due to either the nature of the asset being protected or the demographic of the audience. But you're not going to see your everyday ecommerce, social media, or even banking sites changing en masse.

"Now, one more thing," he says. "If I don't mention biometrics and WebAuth, they'll continually show up in the comments anyway." And so he talks about them and how there are some emerging standards, but he acknowledges they're many years out yet. Anyway, he concludes: "This is why [insert thing here] is not a password killer; and why, for the foreseeable future, we're just going to have to continue getting better at the one authentication scheme that everyone knows how to use: passwords."

Now, okay. So that's what he said. And I just wanted to address it head on.

Leo: Yeah, but you could use the same words about SQRL. I mean, it's not out yet, so it's not like anybody could have adopted it.

Steve: Correct.

Leo: We do have somebody with a license plate "SQRL," however. So you've got that going for you.

Steve: Yes. Our Picture of the Week shows the back of a Jeep with an Arizona-registered license plate, "SQRL." I don't know what the...

Leo: You don't know this guy.

Steve: Don't know who that is. But bravo.

Leo: SQRL.

Steve: And so I will admit that I got a kick out of the fact that the very first reply posted to Troy's article was from a David A. Leedom, 15 days ago, who just wrote: "Has anyone looked at SQRL?"

Leo: Thank you.

Steve: Yes. And there was pretty much, I guess maybe, I don't know, half of the dialogue, there was a lot of dialogue, were people going back and forth. And actually some online names I recognized from GRC's SQRL newsgroup and Twitter were present. The point is, I think, skepticism is a healthy trait, and it doesn't offend or annoy me in the slightest. But what I have also observed is that anyone who has actually seen SQRL and experienced it is immediately and irrevocably converted into a true believer. I think that what confuses people is how patient I am. I deeply believe in the truism that we only get one chance to make a first impression. So I'm working as hard as I can, day and night, so that everyone's first impression will be all that it can and should be.

When I first talked about this on Security Now!, there were a bunch of people who didn't pay close attention, who created blog posts that attacked it, not because there was anything wrong with it, but just because they didn't know what it was, or sort of like this. It's like, well, [grumbling sounds].

So anyway, a few points to Troy's article. First of all, as we know, it's not a product. I think that's probably why it has a chance to succeed. It isn't something you have to subscribe to or pay for or anybody is making any money from, which frankly boggles the minds of friends who I talk to about it when I say, no, it's free. No, I'm just doing this because someone has to so we'll have it.

So as we know, it's a solution and a protocol which by design no one owns. I have simply built an implementation of a SQRL client and now several implementations of SQRL servers. That was done to work out all the details and the edge cases. If I were to do it again, I would likely have built my SQRL client as a web extension, since SQRL really should be integrated into every web browser. Someday, if it succeeds, it will be.

But back then web extensions didn't really exist, and they weren't unified when I began this. And there is a web extension in the works by someone else for Firefox and Chrome. And as for no one owning it, there is also a working Android client over on GitHub whose SQRL implementation is currently available in Arabic, Chinese, Dutch, English, French, German, Hebrew, Japanese, Norwegian, Russian, Spanish, and Swedish. It's working great. There's a native client coming for Linux, and there's a working client for iOS. GitHub also has a crypto library for SQRL that people can use to build into their own apps. And there are server implementations for Java and PHP in the works.

And I'll note that all of these other people are not nuts. They're not crazy. They have one thing in common, which is what for the moment separates them from the rest of the world. They have seen SQRL work firsthand, and they immediately realized what it means. And I know I'm teasing everyone, but this has been a passion of mine for years. I'm beginning to get excited about it because it won't be long now until everyone will have the chance to play with it for themselves and see and judge what this might mean. And as for replacing what we have now, SQRL is quite happy to coexist alongside usernames and passwords, to simply be there as an option and benefit offered by websites who choose to offer it.

And like at any website, if someone is a SQRL user, having created a SQRL identity, and wishes to associate their SQRL identity for that site with their existing identity at a website, they can easily do so. The demos that I have online and which everyone who's

listening to this will be playing with before long, allows you to experience that. You can use a username and password to create an account at this demo site. And then it says, oh, you don't have your SQRL identity associated. And so you can do that, and then you log in with SQRL.

So this has all been designed to allow SQRL adoption to occur incrementally. And, for example, I used this a long time ago, but it still holds. How many times have you read someone's blog, and you've thought, ooh, I have something to add to that. And so you go to click on Reply, and the first thing it asks you to do is create an account. What's your email address? And you look at it, and you think, oh, forget it. No, I'm not. Not one more place on the Internet. It's not worth it.

Well, there's a perfect example of where friction, account creation and login friction, authentication friction is a limiting factor. For those sites to have SQRL means you don't need to do anything except click on the "Login with SQRL," and you are now uniquely identified at that site, now and forever, never having to do anything else. So my point is I believe there are places where account creation friction matters enough that adding it, which is easy to do, will be enough. And what'll happen is, as people incrementally begin to get used to it - and, I mean, what you ask yourself when you experience this is, wait a minute, what just happened? This is secure? And the answer is yes. There's no more usernames and passwords being breached and lost and getting out in the wild because SQRL doesn't give web servers any secrets they have to keep. There's nothing that they have to do. So, I mean, it really is a completely different solution.

So anyway, I agree with Troy 100% that nothing yet has been good enough to supplant usernames and passwords, and that they are here, usernames and passwords, because they've always been. And I also agree that any real meaningful change takes time. We all know about inertia. That's the lesson we're learning on this podcast all the time. IPv6. How many decades old is that? And it's still trying to happen. And TLS 1.0, 20 years old, and it's only now beginning to get itself deprecated.

So anyway, I guess my main point is that until and unless there is a perfect alternative available, just available, and where "perfect" in this context means it's open, it's free, it's zero-friction, and it answers every problem, which I believe we're going to see SQRL does. Until such a thing exists, nothing will ever change. That change cannot be forced. We know that. I agree with Troy 100%. And I would be happy to let SQRL speak for itself. I think we're going to find that it will be able to. And I will be surprised if it doesn't happen.

On the other hand, I'm looking forward to getting back to SpinRite, once this thing is launched and out of my hands because I want to get back to SpinRite. I've had enough of SQRL. And really, I don't have a horse in the race. I just wanted to solve the problems. And I think we're going to see that they're solved. So we'll see.

Leo: I think a lot of people's questions about it just really come from at this point not really having any idea how it works. I mean, we've showed it.

Steve: Yup.

Leo: It's just, even myself, I keep going - on and on. But soon. Soon. Soon.

Steve: Yes. Yes. Yes.

Leo: Anything that makes authentication work will be much improved. Much improved.

Steve: Well, and I wouldn't be surprised if things like password managers, LastPass could easily adopt, I mean, it's a perfect place to add SQRL support. So for sites that don't yet support SQRL...

Leo: That's where you've got to go, yeah.

Steve: You use LastPass, and let LastPass be your SQRL client for what I imagine will be an increasing number of sites that will adopt it over time. Again, it's not an all-or-nothing. It doesn't have to, you know, no one's trying to make any money on this. It has to be free. But it also has to be done right, or it'll just stumble, as other solutions have. I don't think this one will. We'll see.

Leo: I'm rooting for you. That's the case. I can't wait to get rid of passwords. I think that's a good - actually, that's an interesting implementation. If it goes in your password manager, then SQRL gets used wherever it's available. And over time your password manager shrinks.

Steve: Yeah.

Leo: It gets simpler and simpler, yeah.

Steve: Yeah. It just - yup.

Leo: Does SQRL allow for or require two-factor? Or is it just...

Steve: No.

Leo: It doesn't need it.

Steve: It doesn't. And that's the magic. As I've said, the only reason we need multiple factors is none of them by themselves are secure.

Leo: Right.

Steve: So what SQRL is, it is secure single-factor. That is, I mean, and it's not - it's both username and password. That's what's freaky. You don't have to tell it who you are, then prove who you are. It asserts and proves your identity all at once. So, I mean, I showed it to Lorrie because I just, on Saturday, I brought up this new demo site. So I had the iOS client on my iPhone X, and I brought up the site on an iPad, and it was just sitting there. And I let the phone see the QR code, and then I let the phone see my face, and the page on the pad went blip, and I was logged in.

Leo: So cool, yeah.

Steve: And she said, "Why wouldn't anyone want this?" I said, "I know." Well, for one thing, nobody has it yet. I mean, no one has had a chance to because it needed to get done right. And also remember that the QR codes, that's only for optical - to use your phone as the authenticating device. Most people will just click on the "Login with SQRL" on their computer, and so there's no phone involved. Anyway...

Leo: It's basically SSH login, the public/private key login that I use on my SSH server; right? Basically?

Steve: No. No.

Leo: No? Okay.

Steve: No. No. I mean, it's...

Leo: It's more than that. Okay.

Steve: Well, I mean, if you lost your key, if someone got a hold of your SSH key, you'd be screwed.

Leo: My private key, yes.

Steve: Yes. With SQRL you're not. You can get it back.

Leo: Oh. All right. That's cool.

Steve: Oh, I mean it's, you know, again, what we'll do is I want to do a confrontational, you know, you, Mike Elgan, and Jason probably, or whomever. John, if he wants. Anybody. Have a little roundtable and have you guys - I'll show it to you. I'll explain it to you. And then hit me. Because there is an answer for every possible but what about, but what about, but what about.

Leo: You know what I want? You know what I'd like to do is maybe get Troy in or Joe Siegrist, somebody like that. Because honestly, what we really have to do is just get the folks from LastPass to implement it, and it would just take off. They're going to say, but we don't own it. Well, that's right. You don't own passwords, either.

Steve: Exactly.

Leo: That's right.

Steve: Exactly. But if you want a nice integrated solution, I mean, they could say, if they see the handwriting on the wall, it's like, wait a minute, our only chance not to lose people forever is to be offering the alternative at the same time.

Leo: Plus it would help the transition from - because a password manager would continue on legacy sites to unlock the password.

Steve: Yes. They're never going to go away, ever, ever, ever. I recognize that. And in fact one of the really cool things, Leo, is once you're comfortable with SQRL, in the UI you can set a flag, an option button that says request only SQRL authentication or something. I don't remember the exact jargon. But the idea is that then, once you understand how SQRL works, as you visit sites and use SQRL there, the site sees the request saying shut down non-SQRL authentication. So your username and password no longer works there. But not only for you, but for anybody else. That is, it locks the bad guys out. Remember that having SQRL authentication, one aspect is that it's a convenience. But we also want it to be much more secure. Well, the only way for it to be much more secure is if the old crap that isn't secure goes away. And we know that username and password isn't going to go away. But SQRL can ask that it be disabled. And so sites that want to fully support that can, and increase their security. I mean, it's just - it's full of goodies.

Leo: Well, soon; right? Soon; right?

Steve: We'll get there. We'll get there.

Leo: I understand your reluctance. I think you actually for the first time have really clarified that it's because you only get one chance for a first impression. You really want to make sure it is locked down.

Steve: Yes. Because you can't...

Leo: And absolutely perfect.

Steve: You can't come back and have people go, oh, well, remember? Oh, yeah, we fixed that. It's like, unh-unh, no.

Leo: That makes a lot of sense. All right, Steve. Another great show in the can, as they say. Thank you for joining us. We do this each week on Tuesdays, 1:30 Pacific, 4:30 Eastern, 21:30 UTC. You can watch live at TWiT.tv/live, watch or listen. We've got streams, both audio and video. You can also chat in the chatroom at irc.twit.tv. Those are the people watching and listening live at the same time as you. It's a nice community in there.

If you want on-demand versions, Steve's got them at GRC.com, that's his website, the Gibson Research Corporation, along with SpinRite, the world's best hard drive recovery and maintenance utility. If you want to know more about SQRL, it's all

there, too, along with a ton of other free stuff Steve gives away. It's very simple, just go to GRC.com. He also has transcripts of every show there.

We have audio and video at TWiT.tv/sn, and you can of course subscribe in your favorite podcast application. That way you'll get it automatically. I think this is one show, unlike any other show we do, where you want to collect every episode. Get them all. Keep them because it's a library of good information. Thank you, Steve. We'll see you next time on Security Now!.

Steve: Thank you, my friend. Bye. Happy Thanksgiving.

Leo: Thank you.