Listener Feedback #242

Description: Leo and I discuss an oh-so-subtle side-channel attack on Intel processors, the quest for verifiable hacker-proof code (which oh-so-subtle side-channel attacks on processors can exploit anyway), another compiler optimization security gotcha, the challenge of adding new web features without opening routes of exploitation, some good news about the DMCA, Matthew Green and the DMCA, and how the relentless MPAA and RIAA are still pushing limits and threatening the Internet.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-584.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-584-lq.mp3

We talk about the secure ProtonMail service feeling the frightening power of skewed search results, how to regain control over Windows 10 upgrade insistence, a new zero-day vulnerability revealed by Google before Microsoft has patched it, a bit of errata and miscellany, and as many listener feedback questions and comments as we have time for.

SHOW TEASE: It's time for Security Now!. Steve Gibson's here. We have a lot of security news, but we also have a Q&A coming up. We're going to talk about security flaws in Windows. We're going to fix some errors and security flaws in Linux. And I'm excited - our Security Stumper of the Week is back. Stay tuned. Security Now! is next.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 584, recorded Tuesday, November 1st, 2016: Your questions, Steve's answers, #242.

It's time for Security Now!, the show where we cover your security and privacy online. We protect you, thanks to this fellow right here, our Commander in Chief.

Steve Gibson: We protect you, we confuse you, we worry you, and probably upset you.

Leo: Yeah.

Steve: It's all in the name of understanding what's going on.

Leo: That's Steve Gibson, by the way. Let me say your name.

Steve: Oh, yeah.

Leo: And I want to say one other thing, by the way. I don't know if you know, but I was back in Gainesville, Florida on Friday for the ITProTV grand opening. And so that is a bunch of geeks; right? There's, like, 300 people who work for ITProTV, who are customers of ITProTV, and then people who just said, "Hey, let's go see it." And to a person, they said, "Say hi to Steve." They are massive Security Now! fans. And I think that what is really clear is, because this is the geekiest show by far that we do on the network, that these people, who are the geekiest listeners we have on the network, love this show. So you have a very devoted fan base out there, and they all said, "Say hi." At one point I said, "Look, I'm going to have to get a card and start writing this down because I can't keep track." They love you, Steve.

Steve: Well, thank you. Thanks for sharing that. I had a little note in the show notes today to mention that I saw a very nice tweet from Victor van der Veen. He was the project lead on our topic for last week, the Drammer attack, at UV Amsterdam. And he tweeted, "This is so cool. Your explanation of Drammer is absolutely perfect. I just fell in love with your show."

Leo: Aw, isn't that great?

Steve: So, yeah.

Leo: Yeah, and I knew that because I didn't understand a word of it. So I knew that it must be right.

Steve: Well, so nominally we have a Q&A, although we also have just a crazy news week. But I thought, okay. Instead of just doing too much news, I've got 10 questions and answers. The last one was a listener-provided puzzler. So we will definitely at least do that.

But so we'll do the news. It turns out there's - oh, there's just some delicious stuff here - an oh-so-subtle side-channel attack on Intel processors, the actual hardware. Many people have shot me a note, actually it was a couple weeks ago, about the quest for verifiable hacker-proof code, which of course is a little bit of a hobbyhorse of mine because I've always represented that software could be perfect. It's just not. But, you know, it's math, and so it could be, although this little side-channel attack on Intel processors is very worrisome.

We've seen another instance of a compiler optimization wrecking security; an interesting thread about the challenge of adding new web features without opening routes of exploitation; some DMCA news, some good news relative to some exceptions which have just been added which have a two-year life before renewal; Matthew Green asking the court for a formal waiver so that he can write a book; the MPAA and RIAA really going even further than ever in pushing the limits on their language. ProtonMail suggests that

they were almost put out of business by Google, which is sort of interesting. Father Robert, as our listeners know, while you were on your last vacation, Leo, gave up on Windows 10 when it insisted on upgrading...

Leo: Oh, I didn't know that.

Steve: ...in the middle of a podcast.

Leo: Oh, man.

Steve: I have the solution for that. So we'll share that. There's a zero-day vulnerability which Google had to, following their own rules, release, even though Microsoft has not had time to patch it. We have a little bit of errata, some miscellany, and however many questions and feedback from our listeners that we have a chance to get to. Oh, and a wonderful Picture of the Week. Somebody sent this to me, snapped it off of their Windows 10 machine. So lots of fun stuff.

And I need to correct myself. This may not be Windows 10. This is definitely Windows Update, though. And it looks like it's a Windows 7 screen, when I was looking at it while you were talking. Anyway, for those who don't see the show notes, the screen was a snapshot. It says: "Configuring Windows Updates. 4002% complete."

Leo: Oh, lord.

Steve: "Do not turn off your computer."

Leo: It does look like an old screen, huh?

Steve: But wait a minute. If it's 4002% complete, it's really complete.

Leo: It's more complete than not.

Steve: So I don't know why you can't turn it off. It's complete till, you know, 2020, when no more updates will ever be offered for Windows 7.

Leo: This is, though, a category of bug we've had ever since we've had computers, ever since - remember you start copying a file, and it says we'll be done in 342 years?

Steve: I love it, yeah.

Leo: Because it's just doing math; right? And, you know...

Steve: Well, and SpinRite has the same problem because SpinRite estimates how long it'll take to do the whole drive based on how long it's taken to get as far as it has.

Leo: Exactly.

Steve: And many of the problems are at the front of the drive because that's where the heads spend most of their time, so there's more damage that accrues there. And so many people say, Steve, this thing says it's going to take 324 hours. And it's like, no, no, no, no, it's just having a hard time at the beginning so it extrapolates that, assuming the whole thing is going to be that bad. But as soon as it gets past the rough patch, it'll just cruise along. And I often hear them come back and go, oh, yeah, now it's only four hours.

Leo: Although, and I'm sure somebody's going to point this out, I don't know how you get to 4002%.

Steve: Now, that's a stretch. That's not a rounding error.

Leo: That's a typo is what that is, I think.

Steve: Okay. So there's a really big story that would not fit in today. But I wanted to let all of our listeners know that I'm aware of it, and we will do it as the topic next week because it's really interesting. So normally I start off the show notes with "this week on Security Now!." So my second item here is "next week on Security Now!" is the Windows "Atom Bomb" exploit and attack.

Leo: Is this the one Google is talking about?

Steve: No.

Leo: No, all right.

Steve: This is different. This is really a problem because it isn't leveraging a bug. It is leveraging a longstanding feature known as the "global atom table," which is a way for applications to have Windows store arbitrary strings which they then receive a token for. The problem is it's shared. And some very clever hackers figured out how to turn this into an exploit. And here's the problem, is it's not a bug. And Microsoft can't change it.

Leo: Ooh.

Steve: Because it's in all Windows OSes, and existing software depends upon it working just the way it does. So unless something else happens, it will be our Election Day topic for Security Now! - that is, the U.S. election, of course, which is one week from today.

Leo: Wow.

Steve: For next week.

Leo: I can't believe we're in November already. Whew.

Steve: I know. So, which is your birthday month, Leo, at the other end.

Leo: Oh, don't...

Steve: The other end of this month.

Leo: Oh, it's not a good birthday. I don't want to celebrate.

Steve: I had already purchased a hat brush for you by the time I found...

Leo: No.

Steve: I had, by the time...

Leo: No, I'm sorry.

Steve: I was watching you click "Buy It Now," thought, well, I should have known. It was only $14.

Leo: I realized - I got a black hat, and it gets a lot of lint. You are so kind.

Steve: I got the good one for you.

Leo: Yeah, see, you got the good one. I got the $4 one. You are so kind. Thank you, Steve.

Steve: I'll bring it up with me on the day after your birthday.

Leo: I can have it at - it'll be my work hat brush.

Steve: That's right. You need one in both locations. Although I like your Sunday hat better than this one. I really thought...

Leo: This is the flat brim. You like the Sunday - the Sunday hat had the press - I put the little press pass in there.

Steve: Yeah.

Leo: And it really looked like a fedora, didn't it.

Steve: You were having fun with that.

Leo: I like hats.

Steve: So, okay. So I did want to mention something that I only saw after we stopped recording the podcast last week, which is, on cue, Windows Update on my Win7 machine had an optional update which was called "October, 2016 Preview of Monthly Quality Rollup for Windows 7 x64-based systems." Meaning, and this is what we were talking about, part of this new change that began in October is that we will now start getting a single monthly quality rollup, as they call it, on the second Tuesday. And then a week later they are planning to release the preview for the following month.

And it's interesting because there has been a lot of concern, and I teased it at the beginning of the show, about the fact that Windows 10 has become really insistent about installing updates, where at some point you are unable to tell it not to. You're not able to reliably schedule them as Microsoft suggests. I mean, it becomes increasingly insistent. So I'm wondering if the deliberate release of a preview before the recommended, that would allow people, like corporations, the Enterprise users to experiment with it before it arrives. Anyway, I just - this is the first time this has ever happened because this was a few weeks into October. So I did want to say that, as we were told, and as we discussed on the podcast, this is happening. So at the beginning of the month, the second Tuesday, you actually get the single blob containing all the updates for all of the fixes. And then a week later they say, okay, and here's what we've got for next month.

Now, I'm wondering, though, because for example we will be talking shortly about a zero-day vulnerability which Google found both in Flash and in Windows; but, due to Microsoft's release schedule, they weren't able to respond in seven days. Google's policy is we will notify, we will go public, seven days after responsible disclosure, that is, private disclosure, whether the problem is fixed or not, for active, in-the-wild, zero-day exploits.

So now I'm wondering, wait a minute, what happens in this case where Microsoft has issued a preview of their single blob? And then, by the time it comes due, essentially, like comes time to actually deploy it, they change their mind because of something exactly like what has just happened where they, like, oh, shoot, you know, we've got a bad problem we have to address as quickly as possible, where it's possible to address it

quickly. So I'm a little - kind of like it will be interesting to see what happens, whether they disavow this preview of next month's quality rollup or go with it. I just don't know what they're going to do. So it's, you know, it'll be interesting to see.

And I mentioned Victor van der Veen's tweet saying that he had listened to the podcast, and I got the stamp of approval from the guys who did the whole Drammer project. He also subsequently tweeted: "We now know that LPDDR4 RAM is also vulnerable." And he quoted Ars Technica's Dan Goodin, tweeting: "Bit flips on Google Pixel XL, OnePlus 3, Galaxy Note 7...." And I should mention also that, since last week's podcast, I've had a few, but not a huge number, but some people finding both positive and negative results to bit flips. There was one I just saw this morning on a Galaxy Note 5, I think it was. So they do exist, and I'm glad that these guys brought this to light because this is a problem that needs to get fixed.

Okay, now, speaking of problems that, oh, that can't get fixed, or that don't have an obvious fix, this is beautiful. As I described it, "an oh-so-subtle side-channel attack on Intel processors." Now, we know that side-channel attacks are typically clever ways of observing the state or obtaining leaked information from something that you want to keep secret, where somebody figures out that there is, for example, a secret dependency, that is to say, say that a cryptographic algorithm has a secret key. Well, you want the encryption and decryption, for example the timing and power consumption, not to be dependent upon bits of the key. So, and what that would mean is that somebody observing the timing or the power consumption would not be able to determine what the key bits were from something not directly related, but very indirectly related. So that's a side-channel attack. And, you know, and we've covered many of them over time, things like just the sound of the drive heads moving or acoustic noises being generated by fans.

Now, in those cases they were deliberate exfiltration approaches for deliberately essentially telegraphing a secret out of a machine. What these guys found is more of a classic side-channel attack. Researchers at UC Riverside and the State University of New York at Binghamton have devised a robust technique. I think they said it takes, like, 50 milliseconds, I mean, like no time at all to bypass the important protections provided by ASLR, Address Space Layout Randomization. And we've talked about this before.

ASLR is an important technology because what it does is it takes pieces of the operating system, typically the kernel, and places them at boot time in unpredictable locations. This is important because one of the exploits which has been developed to thwart the execute-only or no-execute bit on systems, where for example exploit code cannot bring its own code in and run it on the stack as it used to be able to, it can't do that now because the stack is marked for data only, no execute. So then what happened was the bad guys said, oh, well, obviously the kernel, the OS itself is executable. So if we can find little bits of code and knit that together to achieve our goals, then we can do that. So that happened.

And then we said, okay, let's randomize where the kernel chunks go so that blinded exploit code would not know where to jump to in order to get their work done. So address space layout randomization was again a mitigation against this particular type of problem. These guys have found an incredibly subtle and clean way around it. And if left unfixed, and it's unclear how to fix this, it would render malware attacks much more potent, essentially neutralizing the security benefits of address space layout randomization.

So this was described a couple weeks ago in a paper at the International Symposium on Microarchitecture titled "Jump Over ASLR: Attacking Branch Predictors." And when I saw

the title, it's like, oh, my god. Okay. So we talked years ago, we did a series on this podcast on sort of walking through the evolution of processor architectures from the very early days, like those machines that are over my left shoulder, to where we are today. And toward the end of that series we talked about the ways in which the pressure to increase performance had made the Intel architecture really complicated.

So Intel was struggling against the risk threat. Intel is a CISC, a Complex Instruction Set Computer. So one of the things that Intel did was they read ahead of where the instructions are being executed and look for things that are safe to execute in parallel. And that means that the instructions downstream that the system technically hasn't gotten to yet are doing things that do not depend upon the results from the instructions that are currently being done.

And so this is known as pipelining. And one of the problems with pipelining - oh, and also instruction prefetch, and there's a prefetch queue where all this goes into. One of the problems, though, is when you hit a fork in the road. That is, you come to a jump instruction in the future that you haven't actually gotten to yet. Well, the jump instruction is almost always, inherent by definition, dependent upon a condition, that is, it's called a "conditional jump," on a condition that was left over from an immediately previous instruction.

So pipelines have a problem. They could take both paths. But if they could be smart enough not to take the road less traveled, but to take the probable path, then that would dramatically increase their efficiency. So what Intel did was they incorporated branch prediction. Down in the microarchitecture, down in this incredibly complex chip, there are branch prediction tables which remember, I mean, and even statistically, what the likelihood of individual branches are in the code.

And, for example, analysis has shown that branches tend to go in the same direction. Branches are normally not taken 50/50. For example, in a loop, you might loop 100 times, and then at the end you don't branch back up to the top. You continue going down. Well, that means that that particular branch is 1% not taken, 99% taken. So if there was a system in the processor that could dynamically learn about that branch being taken and not, then it could dramatically increase performance because it would never go reading downstream on a whole bunch of instructions ahead that would end up being thrown away when it branched back up to the top of the loop.

Okay. But think about what that does. That inherently creates a context of what the processor has been doing, and it is inherently shared by everything in the machine. This is just a beautiful insight by these people. So, and again, this as much as anything demonstrates the degree to which almost invisible, but still existent, subtleties can be leveraged into an exploit. What these guys did was they realized that Intel has a certain amount of granularity to the branch table, and that the branch table is shared. So they're able to deliberately set up in their own code subtle tests which allow them to indirectly probe the branch table in their code, which is shared with the kernel. And knowing what the kernel is, and they did this in Linux although it's applicable to Windows and OS X, they're able to essentially deobfuscate the address space layout randomization simply by looking at little fluctuations in the branches their own code takes because that's influenced globally. It's just - it's breathtaking.

So now the problem, of course, is how do you fix this? So these researchers demonstrated the technique on a computer running a recent version of Linux on top of the Haswell processor by exploiting this - they call it a "flaw." It's not. It's a design feature. I mean, they turned it into a flaw in the CPU's performance accelerating branch predictor. The demonstration application that they developed was able to identify the

memory locations where specific chunks of code reside. And it's applicable to virtualized environments, which as we know are common in cloud-based computing and hosting.

And it's not clear how you fix this. Right now the branch prediction table is in the processor, and it's global. That is, it doesn't make any distinction among processes or among privilege rings. So one thing you might do is, if you wanted to protect the OS, is to make separate branch prediction tables for different privilege levels so that applications running at privilege zero, ring zero, would not be sharing the same branch prediction table as those running at ring three. That's a simple thing to do.

In their paper that I have linked to in the show notes, they suggest other sort of workarounds where they would be hardware enhancements. And that's the problem. We know that Intel has themselves a big pipeline of processor architectures for things coming, none of which probably take this into account. So, first of all, it's not clear at all how you fix this on any of the hardware that exists today. And as I mentioned, they demonstrated a robust exploit that takes 50 milliseconds that they set up and run in order to deobfuscate address space layout randomization. So just another beautiful piece of work. And it demonstrates how difficult this, I mean, how difficult security actually is.

And I immediately put the following story, this concept that Quanta Magazine had. They had a story talking about this notion of hacker-proof code. The title was "Hacker-Proof Code Confirmed." And so just to summarize it, this is actually workable, provably correct code which has been shown to function in the real world. From the article they said: "In the summer of 2015 a team of hackers attempted to take control of an unmanned military helicopter known as Little Bird." So this is a helicopter drone.

"The helicopter, which is similar to the piloted version long-favored for U.S. special operations missions, was stationed at a Boeing facility in Arizona." So this was legitimately done as a test. "The hackers had a head start. At the time they began the operation, they already had access to one part of the drone's computer system. From there, all they needed to do was hack into Little Bird's onboard flight-control computer, and the drone was theirs.

"When the project started, this 'Red Team' of hackers could have taken over the helicopter almost as easily as it could break into a home WiFi. But in intervening months, engineers from the Defense Advanced Research Projects Agency (DARPA) had implemented a new kind of security mechanism, a software system that could not be commandeered. Key parts of Little Bird's computer system were unhackable with existing technology, its code as trustworthy as a mathematical proof. Even though the Red Team was given six weeks with the drone and more access to its computing network than genuine bad actors could ever expect to attain, they failed to crack Little Bird's defenses.

"Kathleen Fisher, who is a professor of computer science at Tufts University and the founding program manager of" - get this - "High-Assurance Cyber Military Systems (HACMS)" - I didn't even know we had that, but I'm glad - "said: 'They were not able to break out and disrupt the operation in any way. That result made all of DARPA stand up and say, oh, my goodness, we can actually use this technology in systems we care about.'"

So just to conclude: "The technology that repelled the attackers was a style of software programming known as 'formal verification.' Unlike most computer code, which is written informally and evaluated based mainly on whether it works, formally verified software reads like a mathematical proof. Each statement follows logically from the preceding one. An entire program can be tested with the same certainty that mathematicians prove theorems."

So it's neat that that works. I would argue, though, that while it's necessary, it's not sufficient because, for example, Drammer or this ASLR bypass that we were just talking about. That is, you can have perfectly bug-free code, but now we're beginning to see exploits that no longer depend upon bugs. They depend upon subtle old decisions which nobody worried about once, but are now just - it's like, if you look at a perfectly smooth surface from a distance, if you run your hand over it, it seems absolutely smooth. You look at it under an electron microscope, and it's got bumps all over the place.

So that's sort of what we're dealing with here. We are creating ever-greater degrees of magnification in the way we look at the systems. And everywhere we turn we're finding things that are not bugs, but are just characteristics which, if we really want to, can be used to exploit security. We're sort of in a new era, I think, of depending upon these machines and at the same time discovering that they're less dependable than we were hoping they were.

Leo: I almost feel like they're not less dependable, but that hackers have become amazingly ingenious. I mean, that last, the ASLR hack...

Steve: Yeah.

Leo: It feels like hackers have been so encouraged, or I shouldn't say hackers, security people have become so encouraged by their successes that they've really gotten, you know, it's like I'm looking for a needle in a haystack, but we found a hundred already, so I'm pretty sure I'm going to find one. And they've gotten just really ingenious and clever, much more so than in the past, I think.

Steve: Yeah, I think that's exactly right.

Leo: All the low-hanging fruit's long gone. Now they're just, wow. But it does make you feel like, but it's hopeless, because what are you going to do?

Steve: Yeah, yeah. So here's another example. I got a kick out of this. Now, this was the 15th International Conference on Cryptology. And when I saw it, I said, what a minute. Cryptography? No, cryptology. So they call themselves Cryptology and Network Security. And just to give you a sense for the topics in this, I think it was a three-day conference, we have presentations titled: Diversity Within the Rijndael Design Principles for Resistance to Differential Power Analysis. Or Efficient Implementation of Supersingular Isogeny Diffie-Hellman Key Exchange Protocol on ARM. Or Signer-Anonymous Designated-Verifier Redactable Signatures for Cloud-Based Data Sharing. Or how about Efficient and Secure Multiparty Computations Using a Standard Deck of Playing Cards.

Leo: Oh.

Steve: So, yeah, kind of a fun conference. But I perked up when I ran across When Constant-Time Source Yields Variable-Time Binary: Exploiting Curve25519-donna Built with MSVC - that's Microsoft Visual C - 2015. So as we know, Curve25519 is the one I fell in love with a couple years ago which is the foundation for SQRL. It's an elliptic curve

whose property that I needed was that the private key was deterministic. That is, you could give it a private key which would be used to verify, I'm sorry, which will be used to sign something. And then from the private key it gets the public key, which you then give somebody else to verify your signature under the private key. But the cool thing, the thing where I went, "Oh," I mean, it was the aha moment, was that you could give it the private key. And that allowed me to derive the private key deterministically from a super-secret user key mixed with the domain name so that every domain that you visited would automatically get its own private key.

So, and of course since then, and this was in 2006 when Dan Bernstein presented this, the world has fallen in love with it. We see it everywhere. And so it naturally provides state-of-the-art timing attack protection. Bernstein designed it as a constant-time system. And specifically, as I was mentioning before, in this case particularly the implementation avoids input-dependent branches and input-dependent array indices and other instructions with input-dependent timings - meaning once again that, if you observe it, it isn't the observation of it, where it's timing or power or memory access patterns. Observing it doesn't leak any secret information.

Then Adam Langley, our friend at Google, implemented a constant-time elliptic curve Diffie-Hellman algorithm in C, using Bernstein's Curve25519, calling his version "Donna." So that's curve25519-donna. And he did it correctly. However, as these guys write in their paper: "Although the computation of the scalar multiplication should be time-constant," they write, "we spotted some timing differences depending on the value of the key." And in the show notes I show a chart that they generated where the purple pluses are different keys, and the orange crosses are the same key. And it's very clear that different keys are resulting in different timings than the same key across time.

So they thought, huh. That's not supposed to happen. They dug down into it. And I show the code that was actually generated. And they write: "After a careful analysis of the binary code, the observed timing leakage appeared to be coming from the assembly function llmul" - that's a long multiply - "dot asm found in the Windows runtime library. The function llmul.asm is called to compute the multiplication of two 64-bit integers. It contains a branch condition which causes differences in execution time. If both operands of the multiplication have their 32 most significant bits equal to zero, then the multiplication of these words is avoided as the computation is correctly judged to be zero."

So in the show notes I have a snippet of code. What happened was Adam carefully designed, at the source level, the system to be time-constant. But the library, which was compiled by MSVC 2015, got clever. Whoever designed it realized there was a short-circuit, that is, this was an optimization such that, if the hiwords of these two 64-bits were both zero, why bother doing a multiply? Let's just jump out immediately. And that subtle effect meant that the resulting implementation, while designed and important to be time-constant, actually wasn't in its implementation.

So the short version is Visual Studio breaks time-constant code, which also tells us here's yet another place where we have to really look. It's not enough to design something to be time-constant. You have to verify it, and actually verify it in its final state because you can imagine - and here's the real challenge is that an earlier version, I mean, this was many years go that Adam did this, on a compiler before 2015, where it may in fact have been time-constant; but then, completely asynchronously, an engineer at Microsoft whose job it was, you know, after he had put in 24-bit color into the console so that you could have exactly the kind of cursor color that you want, he then said, "Oh, look. I'm going to optimize the Windows runtime library in assembler." And so he went in and coded this by hand. Anything compiled and linked after that suddenly was broken out of

being time-constant.

So again, another lesson in how hard it really is to get this stuff done. And Leo, I agree with you. It's, I hate to say "hopeless," but, you know...

Leo: That's a subtle bug. I mean, come on. That's a very subtle bug. Geez.

Steve: Yeah. But it skewed the results enough that they were able to extract the key at a distance.

Leo: Wow.

Steve: Yeah, wow.

Leo: Again, I think it's because they've gotten so good; right. The hackers have gotten so good.

Steve: Yeah. It's why we just cannot any longer underestimate what can be done. And in fact in the errata this week, that I'll get to in a second, are two tweets that I quote of people who were annoyed with me last week for they felt overstating the exploitability of the attack on Linux servers, where I was talking about any web-exposed Linux server, and they said this is a local privilege vulnerability only. And anyway, I'll get around to that in a second. But the point is that it's today, but not necessarily tomorrow, because one of our favorite quotes on the podcast is "Attacks never get weaker."

Leo: Mm-hmm, mm-hmm, mm-hmm.

Steve: "They only get stronger."

Leo: Are the bad guys as good as the security guys, though? I mean, these are sophisticated folks, people like Google and some of these companies. And they're white hats.

Steve: Yeah.

Leo: Are the black hats this good? I guess they are.

Steve: I think when, yeah, I think when the payoff is really worthwhile.

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

Steve: Yeah, yeah. Okay. Now we talked a couple months ago about a worrisome API that the W3C, the World Wide Web Consortium, had added into the formal HTML5 spec. And that was, our listeners will remember, the state of the user's battery. And it was, like, seemingly innocuous. But it turns out it was being used to track people.

Leo: Crazy.

Steve: I know, just amazing.

Leo: Ingenious. I mean, you can't knock it.

Steve: Yeah, yeah. So this is sort of like why this is difficult to do. Mozilla's Chris Peterson has a nice thread. I've got the link here in the show notes. But he muses: "What is the use case for the Battery Status API?" which is navigator.getBattery. He asks rhetorically: "Can we remove the Battery API, or perhaps restrict it to non-web content, like browser extensions or privileged web apps?" Because right now any web page that you load with Mozilla, and anybody else who supports this API, can query the status of your battery. And he says: "Chrome and Firefox support the Battery API, but neither Edge nor WebKit have signaled an intent to implement it." And one must think that, now that we're seeing exploits of it, they're not going to change their mind.

So he says: "In theory, web developers would use the Battery API to save document data before the battery dies, to ease off heavy computation when the battery is low, or to implement the Firefox OS settings app. The real-world use cases, however, seem," he writes, "to be fingerprinting users and inflating Uber prices for desperate users with low batteries. Can anyone point," he asks, "to a real website using the Battery API for a legitimate purpose?"

And then he shows some statistics. He says the battery status count probe - now, this is probes built into Firefox, which is sending generic data back to the mothership. You can see the value of this here because it lets them check the real-world usage of things to get some balance. So he says the battery status count probe reports over 200 million battery API calls for Firefox 49.

Leo: Okay.

Steve: And he says the use counter2 deprecated navigatorbattery page probe reports that 6% of web pages use the Battery API. Now, of course that includes ads, remember. So what this really says is that ads have immediately deployed this battery status as another tracking parameter. And they've seen 200 million hits on that. So as he says: "That seems surprisingly high, given the few legitimate use cases." So he says: "I have a patch that makes the Battery API chrome-only" - meaning not web page facing - "and fixes the web platform tests." Then this thread that I linked to contains a bunch of back-and-forth.

I did note that there was some commentary about them having previously reduced the reporting resolution, that is, the number of bits that, who knows, like, what - somewhere there's an A-to-D converter or a down counter or something that was probably creating a very granular, just because it was available, like 16 bits. But nobody needs the lower

eight of those. You know, a half a percent would be a byte because that's 256, or actually even 0.4%. So first what they did was that they fuzzed the results so that fewer bits were available from the API for finger printers. And now they just said, okay, you know.

The upshot of this was nobody has said we absolutely have to have it. And it looks like it's purely tracking. And no one can depend upon it because the biggest browser in the industry, Chrome, doesn't - I'm sorry, Chrome does have it. But Edge doesn't, and WebKit doesn't. So again, a website can't absolutely require it.

And so, sort of stepping back from this a bit, I appreciate that this stuff is difficult. So the steps are the web standards folks add stuff that they think is cool, without really a proven use case; but, hey, let's let the web interface get the status of the mobile battery. Then the browser guys, who want to be fully standards-compliant, implement it so that they can check that off the list, so they're not dinged for not supporting the whole HTML5 suite. Then the real-world folks realize that they have a new tracking hook and immediately deploy it and start using it. Then the browser guys realize that the nifty new feature they put in is being abused to reduce their users' privacy. Now what do they do?

So it's never easy to take things away that have previously been given. But as we know, sometimes that's the best path because this stuff just - there are competing pressures. And of course a classic example is the decision about what to do about certificate authorities that seem to be irresponsible. It's so difficult to take away honoring their certificates. But the system depends upon having reliable CAs. And without enforcement, how does that happen?

So some good news about the DMCA. And this was - I thought it was Iain Thomson, but actually his article is next. So this one was also in The Register. But it turns out that there is a - it was a year late, and this exception coverage was supposed to last three years. But at least it has an additional two years to go before it's revisited. So these are specific exemptions in the DMCA to allow security research. And this, of course, this is a constant refrain on this podcast is, you know, look at the value we are inarguably obtaining from researchers who need to be able to dig into systems that have security in them to verify the security. We have to have that.

So the exemptions cover the use of recorded and streaming video in educational and documentary contexts; the use of electronic literary works in conjunction with assistive technologies; jailbreaking phones and tablets to enable interoperability or remove unwanted software. So that's interesting. That's in. That's an official exclusion now from the DMCA. Efforts to access automobile software. So this sounds like deliberate cover being provided for important classes of research. Efforts to make non-functioning videogames accessible. Efforts to bypass 3D printer materials controls. Seems a little specific, but nice. Efforts by patients to access data in personal medical devices now has an explicit...

Leo: You can hack your own pacemaker.

Steve: ...waiver. Yeah, just don't turn it off. And attempts to reverse-engineer software for security research, broadly.

Leo: We were talking about this yesterday because...

Steve: Oh, good.

Leo: We had Ron Rivest on Triangulation, of RSA.

Steve: Oh, yes, the "R" of RSA.

Leo: Yeah. And he and his colleague David have founded something called VerifiedVoting.org to promote secure voting and to talk about voting and why online voting is a bad idea. And we got to talking about the Diebold voting machines, you remember, from about 10 years ago; and Ed Felten, the Princeton professor who hacked the machines, but was very worried that the DMCA would prevent him from doing that kind of research because he's reverse-engineering proprietary code. They even have, Princeton has a Freedom-to-Tinker.com website. But so I had not realized that that new regulation had gone into effect today, so I didn't - we didn't - I neglected to mention it yesterday.

So a couple of points. If you're interested in election machine fraud and election fraud and how to - there's actually a proposal for how to do a secure election, which of course involves paper auditing. Watch Triangulation from yesterday. It's a great show. And then the second point is, I'm sorry, I didn't realize that that was now legal. You can tinker, thanks to the Librarian of Congress.

Steve: Right. And in fact Ed Felten was mentioned by Iain Thomson. Iain Thomson covered just yesterday a story in The Register titled "Crypto Guru Matt Green" - of course Matthew Green, who we're often talking about - "asks the court for," as Iain put it, or whoever titled his column put it, "DMCA force field so he can safely write a textbook." Iain wrote: "Assistant Professor Matthew Green has asked U.S. courts for protection so that he can write a textbook explaining cryptography without getting sued..."

Leo: Wow.

Steve: I know, I know.

Leo: So sad.

Steve: Like this is the world we're in.

Leo: Yeah, yeah, yeah.

Steve: "...under the Digital Millennium Copyright Act. Green, who teaches at Johns Hopkins University in Maryland, is penning a tome called 'Practical Cryptographic Engineering' that examines the cryptographic mechanisms behind the devices we use every day, such as ATM machines, smart cars, and medical devices." And this is what you want. I mean, what a cool idea for a book. "But this could lead to a jail sentence if the manufacturers file a court case using [the infamous] Section 1201 of the DMCA.

Section 1201 prohibits the circumvention of copyright protection systems installed by manufacturers, and comes with penalties including heavy fines and possible jail time. As such, the Electronic Frontier Foundation (EFF) has taken up Green's case [yay] and that of another researcher to try to get the provision ruled illegal by the courts. EFF staff attorney Kit Walsh said: 'If we want our communications and devices to be secure, we need to protect independent security researchers like Dr. Green.'"

So the EFF has decided to support two actions against the DMCA, Green and this other case of a researcher who wants to make an open-source computer that would allow commercial videos to be edited. EFF is hoping that these test cases can bring down Section 1201, and hopefully the entire DMCA. And I had in my show notes here, God bless the EFF. So I'm so glad we have them as our ombudsmen.

And on the other side we have the MPAA and the RIAA. Oh, boy. Now, this was a piece in TorrentFreak. So, yes, it's got their angle. But it looks like it's mostly facts. There is something known as the Internet Infrastructure Coalition, the I2Coalition, which is a group of it looked like more than a hundred when I scanned through it. But some prominent names popped out, including Amazon, Google, DreamHost, GoDaddy, Plesk, Rackspace, Tucows, and VeriSign, just to name a very few of them. They are urging the U.S. government not to blindly follow the RIAA's and MPAA's new input regarding online piracy threats.

This group is warning that the future of the Internet is at stake. "The problem is those copyright stakeholders are dramatically broadening their attacks beyond specific sites, such as The Pirate Bay, to now include technologies and technology providers which they claim threaten their rights. For example, this year the MPAA and RIAA identified domain name registrars as possible piracy facilitators." A registrar.

In addition, several "rogue," as they called them, hosting providers were mentioned, including Cloudflare. The MPAA characterizes Cloudflare as a service that creates "obstacles to enforcement."

Leo: Oh, my god.

Steve: I know, claiming that it helps pirate sites to hide. So the I2Coalition, pushing back, argues that the submissions, the MPAA and RIAA submissions show a misinterpretation of the obligations domain name registrars have under the Registrar Accreditation Agreement, the RAA.

The MPAA and RIAA would like domain registrars to suspend domain names that are merely accused of copyright infringement. Can you imagine what havoc that would wreak? But most registrars refuse to do so without a court order - rightfully so, according to the I2Coalition. "The vilification," they write, "of technology and misconstruing of the RAA" - that's the Registrar Accreditation Agreement - "have one goal in common: forcing Internet infrastructure companies to act as intermediaries in intellectual property disputes. This is not the answer to intellectual property infringement and is not the purpose of the Special 301 process." That's the process through which the copyright holders are able to make their case. And, they write, "Proposals to extend the use of these companies as intermediaries are misguided."

So, wow. I mean, we all know that every step of the way the RIAA and MPAA, the Motion Picture Association of America, have fought essentially advancements. You know, they didn't want, famously, VCRs. They didn't want home recording of television content, even

for the purpose of time shifting. They absolutely tried to keep it from happening. And what's so weird, Leo, and I know you know this, is look at all the benefit they have reaped from DVDs, which they also fought against. And now the Internet itself. I mean, this is a huge content delivery system. And, yes, there's some piracy. But that goes along with the tremendous benefits that they're getting.

Leo: Yeah. We've been saying this for years.

Steve: Yeah, yeah.

Leo: They never give up.

Steve: So Proton Mail is a service that many of our listeners have asked me about. And I'm, you know, I haven't looked at it closely. It bills itself as a secure email solution. And my problem is that is an oxymoron. So, I mean, I don't know how you do that. And I understand the problem. So it's like, okay. I'm not wanting to discourage anyone from it. I'm sure it's a more secure email solution than essentially no security. But, okay. Anyway, this was really interesting. And in my little note here I titled this: "How Google Almost Killed ProtonMail." And I'll just share what they wrote because it's interesting.

"The short summary is that, for nearly a year, Google was hiding ProtonMail from search results for queries such as 'secure email' and 'encrypted email.' This," they write, "was highly suspicious because ProtonMail has long been the world's largest encrypted email provider. When ProtonMail launched in Beta back in May 2014" - okay, so it hasn't been that long - "our community rapidly grew as people from around the world came together and supported us in our mission to protect privacy in the digital age. Our record-breaking crowdfunding campaign raised over half a million dollars from contributors and provided us with the resources to make ProtonMail competitive against even the biggest players in the email space.

"By the summer of 2015, ProtonMail passed half a million users and was the world's most well-known secure email service. ProtonMail was also ranked well in Google Search at this time, on the first or second page of most queries, including 'encrypted email' and 'secure email.' However, by the end of October" - so that was the summer of 2015. "By the end of October 2015" - so just exactly a year ago - "the situation had changed dramatically," they write, "and ProtonMail was mysteriously no longer showing up for searches of our two main keywords.

"Between the beginning of the summer and the fall of 2015, ProtonMail did undergo a lot of changes. We released ProtonMail 2.0; we went fully open source; we launched mobile apps in beta; and we updated our website, changing our TLD from .ch to the more widely known .com. We also doubled in size, growing to nearly one million users by the fall. All of these changes should have helped ProtonMail search rankings as we became more and more relevant to more people.

"In November 2015 we became aware of the problem and consulted a number of well-known SEO experts," you know, Search Engine Optimization experts. "None of them could explain the issue, especially since ProtonMail has never used any black hat SEO tactics, nor did we observe any used against us. Mysteriously, the issue was entirely limited to Google, as this anomaly was not seen on any other search engine.

"All throughout Spring 2016, we worked in earnest to get in touch with Google. We created two tickets on their web spam report form explaining the situation. We even contacted Google's President of EMEA Strategic Relationships, but received no response nor improvement. Around this time, we also heard about the anti-trust action brought forward by the European Commission against Google, accusing Google of abusing its search monopoly to lower the search rankings of Google competitors. This was worrying news because, as an email service that puts user privacy first, we are the leading alternative to Gmail for those looking for better data privacy.

"Finally, in August, with no other options, we turned to Twitter to press our case. This time, though, we finally got a response, thanks in large part to the hundreds of ProtonMail users who drew attention to the issue and made it impossible to ignore. After a few days, Google informed us that they had. quote, 'fixed something,' without providing further details. The results could be immediately seen."

And I didn't put it in here, but their posting shows their page rank, like historically and on that day, and they just jumped immediately right to the top of the pile again. So I thought that was interesting and just sort of a reminder that Google has an awesome responsibility to be fair, as Google themselves say, to do no evil, or to not be evil. And they do have their own commercial side which is driven by people using Gmail. Yet they're also a search engine. And, you know, I'm now myself aware of the individualization that Google does. Frequently I'm looking at a - I'll do a search, and I'll look at the results that come up, and I'll notice in faint gray, "You've visited this site five times." Which is, you know, it's like, oh, okay.

So we know that, if you're logged in, and Google knows who you are, they're doing what they think is best, hopefully, for you. But I often wonder when I say to someone, oh, just google this and you'll find it, because I do, but that doesn't necessarily mean somebody else will. So anyway, I wanted to shine a little bit of light on this because it's an important thing, I think, that, I mean, we don't know what happened. Maybe, we don't know, maybe there was a problem spidering their site, or they got blacklisted by mistake. Who knows? It's good that it got fixed.

Leo: Yeah. We did that to ourselves with the new TWiT website because we didn't want spidering to go into our old site. So there's a webmaster tool, you can say don't look at the old site. And we didn't know that it's a wildcard tool, and anything to do with TWiT.tv, no matter what the subdomain, will get blocked. So we basically took ourselves off the Google listings. And it took Matt Cutts, who works, worked for Google. I asked Matt, and I said, "What can we do?" And he looked into it, and he figured it out, and he fixed it. So it's easy to do it by accident. So I wouldn't jump to conclude that Google's doing it.

Steve: Right, right.

Leo: And by the way, a little tip, if you want to know what other people see with search results, just use an incognito tab in your browser.

Steve: Ah.

Leo: And you'll be logged out, and you'll see plain search results.

Steve: Good, good.

Leo: Are you ready?

Steve: So, yes. Windows 10 users, high-end users, our listener users, will probably find this interesting. We know, because he said it on the podcast while you were on vacation, Leo, that Father Robert finally gave up on Windows 10 when on two separate occasions it forced an upgrade and reboot that took his system away from him for, like, a chunk of time, right in the middle of a podcast, or when he was doing something that could not be interrupted, like in real-time. And he said, okay, that's it.

So thanks to somebody tweeting, I did some digging, and I found a post, a nicely written post by someone basically ranting about the fact that Windows Update has become user-hostile, and arguing that Microsoft is really disrespecting its users. I don't take any position on that. But he also linked to a different posting which I have in the show notes for anyone who wants it. I'll just walk everybody through this quickly. I mean, I will tell you enough to do it if you're interested, but you can also get it from the show notes. And that is, there is a group policy setting which can be changed to give you back control.

As I understand it, and I don't have this from personal experience, but I've certainly heard of it from many people, Father Robert and the guy who wrote this, who looked at it deeply, it becomes impossible after a certain amount of nagging, and you putting off the upgrade, Window Update finally just decides, look, we're giving you the new code whether you give us permission or not, and it just does it. And there isn't a way to prevent it.

So for everyone except Win10 Home, which doesn't include gpedit.msc, if you just do the Windows "R," or just use the Run option, and then enter gpedit.msc, that launches the Group Policy Editor, which is not a part of Windows most people see, but it's there. And there are all kinds of good stuff. Under Computer Configuration, and then under that Administrative Templates, and under that Windows Components, and under that Windows Update, that will bring you to a right-hand panel that lists a whole bunch of different settings.

What you're looking for there is Configure Automatic Updates. You double-click that in order to open up its settings. And you'll find four different levels which for some reason there's no Level 1, but so they're numbered Level 2 through 5. What you want, I think 3 is the default. You want Level 2, which is notify for download and notify for install. Level 3, if memory serves, I didn't write it, but I think Level 3 was download and notify for install. But that's the default.

And apparently what happens is, if Windows downloads the updates, then at some point it just gets impatient with you, if you just keep saying no, no, not now, it's not convenient now; or if you leave your machine on, for example, overnight because you're in the middle of something, and you wake up in the morning and it's been rebooted without your permission and losing whatever you may have had. So what you want is Level 2, which is notify for download and notify for install. That way Windows doesn't get the download and then become impatient with it.

So you choose Level 2 and make sure you enable the policy. There's a sort of a separate two radio buttons, Enable and Disable the implementation of that policy. Then say Apply and Okay, and you should then regain control of Windows 10. You'll get notified, which you want. But it won't take control from you at any point.

Leo: I think at some point you are compelled to do the updates, though. That was part - if you did the free Windows 10 upgrade, part of the service agreement is that you will do updates. And you know why we want you to do that, because people put off security updates, and they become a hazard on the Internet. So Microsoft made that a condition of the free upgrade. There's a - I'm going to ask Paul and Mary Jo tomorrow. But another way to do that, if you don't have Group Policy Edit, is to - this may have side-effects that you don't like. But it's to say you have a metered connection. Because Windows Update will only automatically download and install if you don't have a metered connection. In other words...

Steve: Oh, that's nice, very nice.

Leo: Right? Because they don't want to incur charges for you on the updates, and these updates are quite hefty.

Steve: Without permission, yeah.

Leo: Right. So if you do that in the network settings, say this is a metered connection, that will also change the behavior. It may have other side effects, though. So I'll ask Paul about this.

Steve: And that is, by the way, one thing I did notice is that the quality rollups are big. I think this month's was 364MB. That's the number that sticks in my mind. The preview for next month is only showing, I just looked at it, 120.5MB. But still, they used to be seven and eight and sometimes 15.

Leo: If you have a small bandwidth cap, and a lot of our listeners do, they don't want to incur charges for you. So I think just say, hey, this WiFi is a metered connection.

Steve: Nice.

Leo: I don't think it'll have any other negative effects. But just be aware that you did that. And do patch eventually. It's just you don't want these annoying - I completely agree with Robert, there's nothing more annoying than trying to do a show, a dialogue.

Steve: Yeah, well, in fact, for two weeks now Skype has said, oh, there's an update. You want to do it? And I go, oh, no, no, no, no, no. Not right now. Let's wait till after the podcast. So today I will remember, and I will let Skype do its update. Actually, I think I

did it last time, so it just seems to be updating all the time.

Leo: There is some urgency. I mean, if you listen to this show, you understand the urgency of applying updates.

Steve: Oh, yes.

Leo: And most people just don't do it. They go, oh, yeah, yeah. So you can see why Microsoft kind of opted for this pushy thing.

Steve: No, and yes, I actually absolutely do understand it. I think that they were recognizing that, I mean, they want to help their users be secure. And you have to push. I mean, look at, you know, Google has done the same thing with security. When they absolutely set a drop-dead for SHA-1 certificates...

Leo: That's why.

Steve: ...we all went, whoa, wait a minute, Gestapo, you know. But it's like, well, that's what was necessary in order to finally get the industry to give up.

Leo: Your mother knew that. Without a good swift kick in the butt, you ain't going to do nothing.

Steve: So speaking of Google, on Friday, October 21st, so what is that, 10 days ago, they reported zero-day vulnerabilities, previously publicly unknown vulnerabilities, to Adobe and Microsoft. Adobe updated Flash on October 26th to address the zero-day flaw that they were given. And Google notes in their security blog: "This update is available via Adobe's updater and Chrome's auto-update.

"After seven days, per our published policy" - and you can see Google's a little sensitive to this. This is why they're making this clear. "After seven days, per our published policy for actively exploited critical vulnerabilities, we are today disclosing the existence of a remaining critical vulnerability in Windows for which no advisory or fix has yet been released." So they gave Microsoft a week, which is not time, you know, not enough time. First of all, I mean, it would have to be, like, really important for Microsoft to, after they'd just unveiled the big bundle updating system, for them to - and as I said, that also seems to create some lead time for them because they're also giving us previews of next month's coming bundle. Suddenly Google says, "Oh, by the way, you've got seven days to fix this." Well, okay.

So anyway, it didn't happen. So they say: "This vulnerability is particularly serious because we know it is being actively exploited. The Windows vulnerability is a local privilege escalation in the Windows kernel that can be used as a security sandbox escape. It can be triggered via the win32k.sys system call SetWindowLongPtr." And they give the details. You give it an index GWLP_ID on a window handle that has a child window, which most windows are. That's anything that is slaved to the parent containing window. "Chrome's sandbox blocks win32k.sys system calls using the Win32k lockdown

mitigation on Windows 10, which prevents exploitation of this sandbox escape vulnerability.

"We encourage," they say, "users to verify that auto-updaters have already updated Flash" - because again this is happening now, Flash is being exploited until this is fixed - "and to manually update if not, and to apply Windows patches" - Leo, to your point - "from Microsoft when they become available for the Windows vulnerability."

Leo: Mm-hmm.

Steve: So as I said at the top of the show, it would be interesting to see how Microsoft dances with this one because this is important. They've already released next month's update preview. I don't know if it incorporates the fix for this. But it seems hard to believe that it could because it was sitting in my Windows Update last Tuesday after the podcast. So, again, there is now an unpatched zero-day in the wild. We don't know anything more about it. It's not remote-exploitable. It's a local privilege elevation. But as Google said, that can be used by an app running on your system to break out of a sandbox.

Which brings me to this week's errata. I got a tweet from a Darko Vrsic, and also Jeff Bearer. Darko tweeted: "Hi, Steve. In SN you said any unpatched web-facing Linux server is vulnerable to Dirty COW exploit. Isn't it only locally exploitable? Jeff was a little harsher. He said: @SGgrc really got his description of Dirty COW wrong. Like super extra wrong."

Leo: Oh.

Steve: He says: "It's not remotely exploitable." Okay. Now, I think the mistake - first of all, they're right. The mistake I made was that you and I were talking about this before we began recording because I had sent you email that morning asking you to bring in some of your Android devices. Although that's different than this vulnerability; right?

Leo: That was for Drammer.

Steve: That was the Drammer. So anyway, I got my wires crossed because I remember there I was talking about how Drammer could be used if, for example, you had a Stagefright exploit, which in fact the Drammer guys did use in order to gain a foothold, and then Drammer was used to give them root access. So these guys are right. Dirty COW is a local exploit. So, for example, it could be used by somebody who had access to a console on a shared system to elevate themselves to root and then get free rein of the system.

The problem is that - and I should have been more clear about this, so I certainly take everybody's point. And that is that it is not itself a remote exploit. But it's worrisome because what we see often is a chain of these vulnerabilities where you link them together in order to achieve what you want. And you just don't want any of these links intact. So everybody's right. I wanted to clean this up. It is not, as far as we know, directly exploitable. But if there was a vulnerability that allowed someone to run some code on a web-facing Linux server, then this allows them to convert themselves to root,

which you absolutely don't want to allow happen. So thank you, everybody, for helping me fix that.

A couple bits of miscellany. I finished Hamilton's "A Night Without Stars," which was a good conclusion to the Faller saga. I love Hamilton's Commonwealth. I love that little universe that - well, it's not little - the galaxy that he has created. But overall I have to say I thought this was not as wonderful as the "Pandora's Star"/"Judas Unchained" pair. It felt a little bit like he was finishing the story because he had to. It was fun; and there were points, as I said last week, when I hit 24% it just took off like a rocket ship. But I'm glad I'm done. I'm not reading anything right now. I'm taking a break. But I just did want to say it was good.

I did want to put on everyone's radar that, except for us in the U.S., apparently, the second season of "Humans" has begun airing. I already saw some tweets about it. We don't get it until February of next year, of 2017, on AMC. And it looks like it's going strong. Season two introduces Carrie Ann Moss, who we all know as Trinity from "The Matrix," as an AI expert who works herself into the plot. And it sort of picks up a couple months after we left off with the first season. And it was only eight episodes. I was surprised when I saw that again because it was so rich, and it was so interesting. So for what it's worth, if you missed it, the first season is available on various outlets, and I think it's really worthwhile. It was really well done.

And I'm not going to say much about this, but this is in the Blue Sky Department. Apparently scientists have actually successfully written data in diamonds. There is a way, by using either a red or a green laser, to flip a molecule in a microscopic flaw in a diamond matrix, and it's a stable change. And it's a long way away from being ready for the world. But I just keep thinking of the little storage wafers on Star Trek; and I'm thinking, you know, we're heading in that direction.

Leo: It would last forever, too; right? I mean...

Steve: It absolutely would. It would be, as they say, "Diamonds Are Forever." And this article, it was in The New York Times, it also said, "And your data could be, too." So again, blue sky, no time soon. And apparently you can use industrial manufactured diamonds. You don't need to have high-end...

Leo: Oh, that's a relief, yeah.

Steve: Yeah, exactly. And I did get - now, this is weird. I thought, what's the punchline? Because he says: "Hi Steve. A SpinRite story for you with a punchline." This is from Matthew Olmsted in Davis, California. And he was sort of stretching a little bit. Or reaching. He said: "My story begins when I advised my dad to give your product a try on a failing drive in his laptop. Sadly, it was already so far gone mechanically that SpinRite wasn't any help, though he was able to eventually get the data off." So I guess he just kept trying, but SpinRite wasn't able to, like, fix the drive.

"He was frustrated and decided to give his SpinRite license to me. Fast-forward a few months, and a friend of mine in Colorado" - thus the punchline we're getting to - "has started having troubles, saying his system was getting blue screens on boot with the error mentioning: 'This was probably caused by the following module: hal.dll.' Which all of us long-time Windows users know is one of the core boot dll's and often causes a

problem.

"After looking it up," Matthew writes, "I found that it's the Hardware Abstraction Layer module. Not good. I sent my friend a copy of the SpinRite ISO and walked him through setting up a bootable CD. After that, I got him set up running a scan over a drive. The next I heard from him, the system was running again without any problems. So whether you like it or not, Steve, you can now say of SpinRite: 'So easy, a stoner can do it.'"

Leo: Hah. Yeah, maybe not.

Steve: I don't think that's going to be my tagline. I think "It works" is probably what I'm going to stay with.

Leo: It works is the main thing.

Steve: Yeah.

Leo: It works. All right, Steve. I've got questions. I'm thinking that since you gave me the questions, you have answers. I could be mistaken. I don't know.

Steve: Generally I do.

Leo: Maybe not for the last one. Actually, I know you do. I know you do.

Steve: Ah, yes.

Leo: You know all. You may not tell all, but you know all. This is a tweet, comes to us from @dpmanthei. I guess that's his name. Would changing the target - this is an interesting question - of my Chrome bookmarks, instead of the name of the site, the actual IP address of the site, would that mitigate the impact of a DNS DDoS? In other words, if you don't need DNS, who cares if it's down?

Steve: Exactly. So, okay. The first problem is that, if a site's IP address changes, obviously then the bookmark would be obsolete.

Leo: Right.

Steve: One of the beauties of DNS is that it does provide a mechanism, as we've discussed, for allowing a controlled degree of caching. But it's easy to look up a site's address in Windows from a command prompt. You do nslookup, space, and then the domain name; and, bang, you get the IP. Now, sometimes you get five because one of the ways that sites load balance or handle redundancy is they will give you a batch of IPs. And some DNS servers will automatically rotate them with every query because the

idea is you will try them in the order that they are delivered. And so that statistically spreads users out across a number of different IPs. And the advantage is, if one goes down, then the DNS record contains four others that the system can try in succession. And it also automatically, as I said, spreads people out.

But let's say that's not a problem, that is, that the IP is fixed. The problem you still have is that, when you obtain the web page, it absolutely contains URLs for other assets from that site. Now, the URLs could be relative URLs, meaning that they just begin with a slash for the root and then specify the directory. If a browser sees that it implies the same HTTP whatever, colon slash slash whatever domain that the page came from. So you can use what is known as a relative URL. But what I tend to see on web pages is explicit URLs, where every other, you know, widget and gadget and thing on the page says https://www dot whatever.

So the problem is you could get the first page. You could get sort of the page framework. But the assets from that site would still need DNS to look them up. Technically, your system knows what the IP is, but it doesn't know about the mapping. So if you were really wanting to do this, that is, to switch your bookmarks to an IP, the hosts file, the hosts file that I talked about as the way I solved that problem that morning by mapping the IP for our upstream merchant provider into the hosts file. Suddenly then the commerce system came back alive because it thought it was able to look up the domain name.

So if you wanted that, you could simply create a hosts file with all of those domains and their IPs at the time that you create the file. And if there was ever an outage like this, then you'd drop that hosts file in, or change the hosts file from your normal mostly empty one to one that's been populated, and suddenly you and you alone in your neighborhood will have access to those sites because you've hardwired the lookups into your local system.

So just setting the tab or the bookmark probably isn't going to do the job because pages will have so many references inside to other assets on the same server. If they were relative URLs, then it would probably work all right. It would just fetch all of those things. I mean, the site might be a little flaky, like if it tried to move you to a secure, if you weren't secure, and it moved you there, that might break things and cause things to stop. So you really can't rely on it. But essentially the hosts file is the first place the system looks. So if you were interested in creating a defense against this, you could create a fully populated hosts file, and it would work.

Leo: Yeah. A lot of people do that. That's also a way to block ad sites, by pointing a known ad site URL or number to local hosts, something like that.

Steve: Right. In fact, at GRC, I have a 10-dot network inside the network. Yet the various machines there need to refer to each other using domain names. But the problem is, if they look up DNS, they get the public IP, which is in Level 3's space, you know, 4.79.142.200, for example, for GRC.com, which won't allow them to reach a machine inside the network. So I use, I mean, it was easy to do because I didn't have many places to do it. I used a hosts file on those machines to override the lookup of public DNS, and so they're able to find each other in the 10-dot network.

Leo: Of course you could just memorize all the important IP addresses. Just type

those in instead of names.

Steve: That's right.

Leo: That's the way a real geek would do it. @Sundance1555 asks: Why don't browsers store/cache IPs to skip DNS? Wouldn't that prevent attacks like what happened on Friday?

Steve: You know, and that question always comes up because we've had attacks on DNS before. And people say, hey, you know, why is the system so stupid? Well, the reason - our listeners probably know that you always see pairs of DNS numbers, DNS IPs. That is, like 4.4.4.1 and 4.4.4.2. Generally it's in pairs. The only reason for that is the original recognition of the importance of DNS. And so the original people who implemented this said, "You know, if we don't have DNS, we don't have anything. So let's always provide a pair of servers." In some cases, they both map to the same hardware. Technically, they're supposed to be in different networks. And you'll see sort of older school providers who remember those days will deliberately have the second to the last digit be different. Even if it's technically on the same network, they're just wanting to play by the rules, which was you ought to have geographically dispersed DNS because it's really important for people to get to. And a local network outage should not bring your DNS down.

So it is important. But that's really the only concession that was made. The original designers said, oh, we'll just give them redundancy. There wasn't this notion of, wow, we're going to be suffering through long attacks without DNS. We need to fall back to the most recently known IP. Why not use that? And in fact what Sundance1555 has suggested is a good idea. And it's been discussed, for example, that it'd be nice to have a DNS add-on or like a local shim that could intercept requests and cache them statically and permanently so that, if DNS fails, and you're unable to obtain a fresh IP address, falling back to what was the last obtained IP for that domain name, that would work. It would certainly work for GRC and for most other companies, just as it worked for my upstream merchant provider.

So it's not something that our systems have at the moment; but, yeah, it's a good idea. And if attacking DNS becomes more prevalent, I bet we're going to see somebody do that because it would allow people to cruise right across an outage without even knowing it.

Leo: Yeah. Robert Hancock, @chuhaitime on Twitter, notes a known bug in the EdgeRouter X: Hi, Steve and Leo. I think Robert's one of the people I saw at ITProTV, by the way.

Steve: Hi, Robert.

Leo: Just FYI, there's a bug in the EdgeRouter X bootloader which causes it to function as a simple switch at boot. A firmware update does not fix it. The file from the Ubiquiti forums, update-boot.sh, it's a shell script, will fix it.

Steve: So this was interesting for two reasons. First of all, I wanted to bring this to the attention of our listeners because...

Leo: A lot of them have it because of you.

Steve: Yeah. This is our favorite router. I mean, it's $49, and it's a five-port router and a little powerhouse. But the problem is an interesting one. The bug is that - and what's so interesting is that it happens actually a lot of the time. When it is booting up, the hardware comes up, and it functions as a switch. And it's only after a supervisory layer or process starts, a little bit after that, that it then becomes a rule obeying router. And the problem is that could be leveraged. That little slice of time could be leveraged for exploit.

For example, say that a high-level state actor wanted access to your network. So they briefly shut off your power and bring it back up and then immediately start hitting your router. They're going to get through because it won't be doing any NAT or any blocking or any rule-obeying. It's a simple switch, giving them complete brief access into your network. I mean, again, it's a farfetched scenario; but what we're seeing now is that farfetched scenarios are being realized.

So although this is a shell script, I grabbed it and looked at it because I was curious. First of all, I was curious when it was so big. I wrote it down, I thought, somewhere. Oh, yeah, 166K. So I thought, okay, wait a minute. What? It's mostly, it is a shell script. It's a little bit of script at the top and at the bottom of a huge 165K base64 encoded ASCII blob.

Leo: Ooh, I don't like that.

Steve: Well, right.

Leo: Because you can't tell what it is.

Steve: Well, but it is, I mean, you would download firmware, right, from Ubiquiti.

Leo: Yeah.

Steve: And they do provide an SHA-1 hash so you can verify that it has not been changed. And it's coming from - the actual shell link is ubnt.com/firmwares/edgemax. So it is coming from them. It's not coming from like a forum posting. So I'm glad you brought that up, Leo, because, yes, it is actually from their firmware repository. But for whatever reason they did it as an update boot shell script.

Leo: Yeah. I like that, actually, yeah.

Steve: Yeah, yeah. It's easy to deploy. So you just use cURL in order to grab it and then run it. And it goes. And it fixes this, which prevents there from being this little gap while

it's booting that it is not a router, it's a switch, because there are some conceivable exploits of that. So thanks, Robert.

Leo: Yeah. And I might have gotten Robert's Twitter handle wrong. It is ambiguous, because [Bleak] tells me that Chuhai is a beverage, an alcoholic beverage in Japan. So it could either be chu-haiti-me or chuhai-time. I think it's ambiguous. Don't know what it really means. Only Robert knows.

Steve: And I was going to say most people have no idea.

Leo: No idea.

Steve: They're just like [crosstalk].

Leo: I like chu-haiti-me. Will Larson in Gilbert, Arizona has deployed Local Egress Filtering, or LEF: Hey, Steve and Leo. Just listening to the podcast about DDOS and Dyn, and you mention egress filtering at the ISP level. I mentioned it, and you told us why it wouldn't work. Using a packet filter in a local router - I use OpenWRT firmware - I have my router check that the source IP is legitimate - oh, this is a good idea - before it leaves my WAN port.

Steve: Ah, yes.

Leo: So everybody could do that, and at least they'd know that they weren't having a spoofed IP address on an outbound packet.

Steve: Now, here's what was interesting. It's exactly that, Leo. I hadn't - either I knew this once and I'd forgotten it, because I was once deep into NAT'ing and all that, or it never occurred to me. But I don't know why a NAT router would pass...

Leo: Right. It should - yeah. I mean...

Steve: Yeah. Because as a packet...

Leo: It's clearly wrong.

Steve: Right. As a packet is leaving, what it does is, because it's called "packet rewriting," it rewrites the source IP from an IP in the LAN to the router's current public IP. That way - so the packet arrives at the router from the inside, from the user's network. And it's got a source IP of one of the machines in the LAN, you know, 192.168 dot something dot something. The router, the act of NAT is it changes that 192.168 that'll be in the packet header, in the IP packet header, changes that to whatever the IP is, four dot whatever, eight, you know, the public IP. And it makes an entry in the NAT

table which says, when a packet comes back from the same destination IP and port, change the destination back to 192.168 whatever and stick it onto the internal network so it'll find its way back to that machine. That's NAT.

But what would the router do when it gets a packet trying to leave that doesn't have a local IP, that is not 192.168, but it's like the IP of some DNS server somewhere, which some light bulb or your toaster or something that's been taken over remotely is attacking. So I'm a little mystified. Do routers, I mean, if it's possible to spoof a source IP across a NAT router, that would suggest - and this is where I'm thinking either I knew it once and forgot it or I never knew it. But that would suggest that the router looks at the source IP, decides first is it within the LAN range. And, if so, it NATs it; and, if not, it passes it.

But if routers simply dropped a packet that has got a spoofed source IP, all of this problem would go away because there's not a light bulb in hell that is not behind a NAT router. They're all behind NAT routers.

Leo: But didn't you tell me last week that that doesn't work at the ISP level because these IoT devices aren't in fact sending spoofed source packets? They don't care.

Steve: Correct.

Leo: They say, yeah, it's me, I'm doing it, because even - there's millions of them.

Steve: Yes, you're right. And that's the argument about anybody bothering with all this is it's not going to completely solve the problem because they simply do legitimate HTTP connections or DNS queries or [crosstalk].

Leo: Or GREs or whatever.

Steve: Yes.

Leo: And the reason I brought it up last week is because in the early days of DDoS attacks they would use raw sockets, which you rightly pointed out is a real flaw, to spoof their outbound address, or I'm sorry, their source address, so they couldn't be stopped. But that was when you had a limited number of machines doing the attack. Now that you have virtually infinite machines, you don't care. You don't care.

Steve: Hundreds of millions.

Leo: Right. No reason to hide. You can't be, you know, you can be blocked, but there's plenty more where those came from.

Steve: Okay. Let's skip to the last question. It's 4:00 o'clock.

Leo: All right. All right.

Steve: And we've got two hours on the podcast. We'll pick up these questions either next week or the week after. But we've got to do our Brain Teaser of the Week.

Leo: And this is a long one. Wow. Holy cow. Or am I looking at the wrong thing?

Steve: Yeah. Way down at the bottom, number 10, James Patrick.

Leo: Way down at the - oh, yeah. Well, I only see five. Oh, there it is. Ah, yeah, yeah, yeah. James Patrick, Indiana, brings us the Security Puzzler of the Week. Listen carefully, friends. I guess you don't have to listen carefully. You can always rewind it if you didn't get it the first time. We know, writes James, that when we have compressible data which we also want to keep private through encryption, if the data is compressible, we want to compress it first, then encrypt it, since encryption creates maximum entropy which is inherently uncompressible. In other words, if you encrypt it first, you wouldn't be able to compress it because it would be data. It would be random anyway.

Steve: Right.

Leo: But can anyone think of a case - and this is the stumper; right? - where compressing the file before encrypting it, compressing before encryption, which is the routine course of events, would have undesirable side effects? It would give you a result you didn't want.

Steve: Yes. Think about that for the week, and I will explain why that could be a problem.

Leo: So you might not want to compress at all.

Steve: The question is how can that be a problem?

Leo: I think I know. Do you know?

Steve: You gave it away last time, so don't guess.

Leo: I'm not - was I right last time? Oh, I didn't even know.

Steve: Yeah, you were.

Leo: So much for that. I think I know. But we'll find out next week. That's an excuse to listen next week.

Steve: That's what they need. They need a reason.

Leo: Another excuse, yeah, they don't need [crosstalk]. Every Tuesday, 1:30 Pacific, 4:30 Eastern, 20:30 UTC, we gather together to sit at the feet of the master and learn. And, boy, is it an education, 584 episodes' worth. And 585 is next week. Now, if you can't be here in person, and it's kind of fun, you know, the chatroom, and we're all sitting around, we're listening together. But if you can't do that, you can always get it on demand. And Steve makes it available at his website. We do at ours, too.

But I'll tell you Steve's first, GRC.com. The reason you might want to go there, he not only has the audio of the show, but he also has a transcript, which we don't have. A human, Elaine, writes that all down, and she does a great job of transcribing it so you can read along as you listen. He also has so many other things at GRC.com that it's worth a trip, not to mention SpinRite, the world's finest hard drive maintenance and recovery utility, which I just mentioned. But things like the Healthy Sleep Formula and the ShieldsUP! and the Perfect Paper Passwords and details on the latest SQRL. There's a whole lot of stuff there, GRC.com, all written in beautiful 1994 HTML. No, you do use JavaScript. Right? Or no, just CSS.

Steve: Just CSS. I know JavaScript. I've written a bunch of stuff, as we know.

Leo: Oh, yeah. But you don't need it.

Steve: I don't need it.

Leo: That's the point. And the whole menu system is CSS.

Steve: I know all my hex colors. I know all my hex colors.

Leo: In his head.

Steve: #990033. That's my favorite red.

Leo: What's your favorite color? Oh, #006699, of course. Don't you know? Steve is also on the Twitter at @SGgrc. And as you could see, a lot of the questions come via that, so you can also DM him there. He accepts DMs from everybody. We have the show at TWiT.tv/sn, YouTube.com/securitynow. It's on every podcast platform out there. You absolutely must subscribe. And what I think most people do, and I certainly encourage it, is archive past episodes. This is not one of those things where

you want to delete after you've listened to it. Keep it around. Burn it to a DVD or something because there's always stuff to go back to, to refer to, and we constantly do that on the show, too. We'll say, well, as you remember from Episode 432, packets are not...

Steve: Well, for example, that series we did on processor architecture was really cool back then.

Leo: It's an education.

Steve: And it's still valid today.

Leo: Yeah, this show is an education. So please take advantage of that. They're all available for you at GRC.com and TWiT.tv/sn. Thanks, Steve. I'm excited we're going to - in a month we're going to see you in-studio.

Steve: Yes, you are.

Leo: You're going to bring me a hat brush.

Steve: I'm going to bring you a hat brush.

Leo: And we're going to do our special holiday episode. And I think we can mention that because I think we're going to stream it live. We're going to prerecord an episode for Christmas week with you and a bunch of other hosts. Denise Howell is coming up for it.

Steve: I think Rene is going to be there.

Leo: Rene Ritchie's going to come for it. Because we had so much fun...

Steve: And Paul.

Leo: And Paul Thurrott. You remember a couple of years ago, and this was the inspiration, during our New Year's Eve thing.

Steve: Oh, great time.

Leo: We got you and Randal, bunch of other hosts, all sat at a roundtable, and we

did half an hour. And it was great because it was so cross-discipline. And you guys don't usually get to talk to one another. I think there's been a drumbeat of you and Paul Thurrott in the same room for a long time. I think that will be - we can fight over Windows 10. Should be a lot of fun. Thank you so much, and I'll see you next time.

Steve: Okay. Thanks, Leo, talk to you next week, on Election Day.

Leo: Ooh.