SECURITY NOW!

Transcript of Episode #410

Interesting & Useful Intel History

Description: After catching up with another post-PRISM week of security industry news, Steve and Leo wind up and release their propeller beanies for a deep dive into the early history of Intel processor memory management - which, it turns out, has direct application to Steve's current work on SpinRite v6.1.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-410.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-410-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Lots of security news, more about the NSA, and then he's going to talk about some of the things he remembers he learned way back when, when he was writing SpinRite, that are still true today. A look at Intel memory management and more, next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 410, recorded June 26th, 2013: Interesting Intel History.

It's time for Security Now!, the show that protects you, your loved ones, your privacy, your security. Man, there wasn't - there couldn't be a better time. We thought when we started doing this show seven years ago that we'd run out of topics.

Steve Gibson: Yeah, and maybe take about 20 minutes a day, or a week, 20 minutes a week, just sort of a quick little touch base. What happened?

Leo: I have a new prediction. Steve Gibson, our host and Explainer in Chief, I have a new prediction. This show's going to get longer, and there's more stuff. In the next 10 years, privacy and security are going to be THE topic of the day. That's my thought.

Steve: It's funny because Elaine quotes for her transcription cost based on the length of the podcast.

Leo: Oh, it's been getting longer, huh.

Steve: So, well, I've tripled my budget for...

Leo: Oh, geez. Oh, man.

Steve: ...for the transcripts that we create from the podcast.

Leo: We'll go in with you. Contact Lisa. We'll split the cost. Steve is here. He is the man at GRC.com, the creator of SpinRite. And that's going to be germane, and you'll find out why in a moment. It's the world's finest hard drive recovery and maintenance utility. He's also a great benefactor to all humankind with the free stuff he puts up at GRC.com - well, all computing humankind - like ShieldsUP!...

Steve: A subset. A worthwhile subset.

Leo: ...Password Haystacks and more. And what are we going to do today, Steve? I see the title.

Steve: Well, okay. So our - yeah. Our listeners know, because I talked about it briefly the last couple weeks, that I'm back at work on SpinRite. And frankly, I was a little nervous about mentioning it. I've actually been working on it for about two months.

Leo: You don't want deadlines.

Steve: Well, I don't. I don't work with deadlines. But I was worried that it would stop its sales. I mean, that if people knew there was a new one coming, they'd say, oh, I might as well wait for that.

Leo: True.

Steve: But the fact is, sales has not been hurt by my talking about it. And it is the case that all - I've made the promise to all existing SpinRite 6 owners that what I do next will be free, that everyone who has it will be able to update themselves to this next one I'm working on.

Leo: Isn't that generous.

Steve: Well, it just, I mean, it's been nine years, and people are saying, oh, we want to give you more money.

Leo: Has it really been nine years since the last?

Steve: Yeah. I finished - it was '04 that SpinRite 6 launched. And not a byte has changed in nine years. On the other hand, not a byte has changed in nine years.

Leo: Right, it didn't need to be.

Steve: Well, it didn't need to be. But over time there have been some things that have come along. For example, now drives often have the so-called "advanced format" with 4K sectors, rather than traditional 512-byte sectors. And so that could interact. I mean, the drives are, like, upward compatible. SpinRite, frankly, has been rather amazingly upward compatible, such that it's nine years old and still is recovering data for people without any problems. But there are a bunch of things that I can make better. And I got all these other things done, and it's like, okay, SpinRite was calling.

So what I'm going to talk about today I think everyone is going to find interesting. It's one of our - we haven't done this actually for years, one of our fundamental technology propeller-head episodes. Anybody who pays attention, I don't know if you could, I mean, you have to be listening to this in order to understand it. But none of this is difficult. And essentially what I'm going to explain is how what seems to be a mistake, an oversight in the early design of the evolution of the Intel processor, from the 8088 that we first got in the IBM PC to the 286 to the 386, right back in that region, something was discovered. And it turns out I need it today for SpinRite, for the next - for the evolution of SpinRite. And I had sort of a hazy understanding of it a week ago. Now I'm an expert. I know exactly what happened. I know how it happened and why it happened and what it does.

And so we're going to go back in time and look. And the old-timers among us will be sort of familiar with all this, but maybe not have ever focused on it as much as I have had to recently. And I think everyone's going to find it interesting, even young kids. It's like, oh, you mean there actually was a history before the Internet? Yes, there was. There was. We had computers before...

Leo: And you and I remember it. I mean, we were there.

Steve: Yeah, yeah.

Leo: You were writing for InfoWorld at the time.

Steve: So that's our topic. And we do have a bunch of interesting stuff, I mean, like at the top of the show, as usual. My very favorite bit of NSA PRISM humor was - I've embellished on the concept, which I got from Twitter, Matt Surabian, I think that's how he would pronounce his name. So it's short, being a tweet, although I did enhance it. So it is: "Okay, NSA. How's this for a compromise? You add packet-level virus detection and removal to your splitter boxes, and we'll call it even."

Leo: Yeah, at least we saw something for it.

Steve: Yeah, exactly. It's like, hey, you add Internet-wide antivirus, and it's like, oh, okay, then we'll go ahead.

Leo: Did you see that...

Steve: Oh, yes. I even know what you're going to say.

Leo: Do you really?

Steve: I bet I do.

Leo: So if you use encryption or Tor...

Steve: Yes. It's in the notes here, Leo. I know. You are presumed guilty.

Leo: Yeah, it's like, oh, we're definitely capturing that traffic. I don't care where you are.

Steve: If you didn't - yeah, well, okay, I'm getting ahead of myself. But so...

Leo: All right, save it.

Steve: But, yes, I did see it. And I was just like, oh, my goodness.

Leo: Unbelievable.

Steve: So since we talked last, another drop of news from our friend - or some people don't like him, other people think he's a hero, blah blah blah. He's certainly controversial. Our controversial entity, Edward Snowden...

Leo: The leaker. The AP Style Guide says you're supposed to call him a "leaker."

Steve: Okay. I think that works.

Leo: Because if you call him a whistleblower, that means you like what he did. If you call him a traitor, that means you don't.

Steve: Now we have espionage, of course. Now he's, I don't know, is there a - are you a something if you are guilty of espionage?

Leo: It's a spy.

Steve: It's a spy, okay. So...

Leo: Yeah. He's being accused under the 1917 Espionage Act. And it's ironic that they're - the U.S. is accusing him of spying for revealing that the U.S. is spying on us.

Steve: Yeah. Exactly. Well, and again, it's that fantastic Stephen Colbert tweet, why are you upset about - and I'm paraphrasing because I don't remember it exactly. But it was like, why are you getting upset about what the NSA is doing with their secret spying program that they're hiding from everyone? So it's like, okay.

Leo: [Growling]

Steve: Yeah. Anyway, so the news was, and many people tweeted this to make sure that I was aware of it, that it turns out the U.K. is engaged in as large, if maybe not even larger, an Internet spying operation. And this time we know that they are tapping all of the fiber optic cables coming in and out. So this was the first time we've seen contemporary - I don't know if this is confirmation of my theory, but many people have felt it was because, in the news that we got that was most recently released, it was, okay, the U.K. is tapping everything. Oh, and they are of course cooperatively sharing with the U.S.

Leo: Which means it's I'm sure reciprocal. It's not like the U.S. would say, please give us everything.

Steve: No.

Leo: Yeah.

Steve: Even though we've probably already seen it. We'd like a second copy. Oh, goodness. And so, yes. And so here in my notes I said "additional clarifications to Congress." This came out during the hearings that have resulted from Edward's leaking. And that is, the NSA's guidelines and regulations state that "encrypted communications," just the fact that they're encrypted, is in and of itself suspicious due to what it might contain and is therefore subject to lawful capture and storage. So if you're encrypting it, that's suspicious in and of itself, so we can save it. So now we do know where all of those zettabytes are going.

Leo: Mm-hmm.

Steve: Yeah.

Leo: And if you use Tor, same thing.

Steve: Oh, yeah.

Leo: Unbelievable.

Steve: So we just had yesterday a new release of Firefox. The "Countdown to 23" is what I call this because 23 is when third-party cookies get blocked by default. And that's still happening. Mozilla is still rattling their sabers, and it's still causing upset and concern. On the 19th of this month the Washington Post in the Business Technology section said Firefox - the headline was "Firefox Browser to Move Ahead with Do Not Track." And actually that was a little bit of a misprint or misstatement because what they were actually talking about was third-party cookie blocking. So that's slated for 23. As we know, it was pushed back until August. And so that should be when we see 23.

Leo: Oh. So you're talking about third-party cookies, not Do Not Track.

Steve: Yes.

Leo: Oh, okay.

Steve: Yeah, Do Not Track is already there.

Leo: So third-party cookies are blocked on Safari by default.

Steve: Correct.

Leo: What does Chrome do? Chrome, I think, doesn't because...

Steve: Unh-unh. Chrome's not...

Leo: ...Google's an advertising company. So they're never going to...

Steve: Precisely. Precisely. It'll be really interesting to me if Chrome is, I mean, as you said, Safari has always been a third-party cookie blocker. Mozilla and Firefox, which is super popular, of course, will join that camp. Microsoft always talks about it and then never does it. Several of the betas have had that on, and then they always backpedal because of pressure from the advertising industry. So that will leave those two, essentially. Well, and Opera, although Opera doesn't have a large market share. So of the two biggies, IE and Chrome, we'll see what happens.

My sense is this is a battle the advertising industry is losing because they're not - essentially, if you look at the dialogue going on, they are refusing to budge. They're unwilling to honor Do Not Track. And so Mozilla is saying, fine, we're just going to block third-party cookies, if you guys won't come to the table and negotiate in good faith. And the advertising industry is saying, you can't understand, we have to track people. And but, you know, they're not tracking Safari users, and they're soon not going to be tracking Firefox users.

Leo: Well, there may be an unfortunate unintended consequence because they've said now, well, we're just going to go to fingerprinting. Which we've talked about before, this ability.

Steve: Yup.

Leo: They don't need cookies really.

Steve: No.

Leo: And the fingerprinting you can't really stop.

Steve: Oh, yeah. Oh, yeah.

Leo: You have to work in incognito mode or something; right?

Steve: Well, we can - no. We'll talk about that.

Leo: Oh, good. All right.

Steve: Yeah, we can block that, too. I just - my take is nothing will happen. This is much to do about nothing. Nothing will happen. Advertising will still work. Revenue will still flow. The whole ecosystem, it does actually not depend upon this.

Leo: Well, also because in all likelihood most people will leave it on, if it's on by - off by default.

Steve: No, no, that's the point. It's going on by default.

Leo: Right, right. But...

Steve: So right now...

Leo: Internet Explorer and Chrome occupy what percentage of the total browsing?

Steve: You're right, they're - Chrome is...

Leo: 80%? 70%?

Steve: Chrome is really making inroads.

Leo: Yeah, so...

Steve: And I don't think IE has ever dropped below 50 because it's just the default browser.

Leo: And if they start using fingerprinting, they're not going to announce it.

Steve: No.

Leo: Fingerprinting, we should mention, is the ability to figure out who you are, not based on any cookie, but just things like browser resolution and just - if you put enough factors together, everybody's unique.

Steve: Well, it's - yes. We've talked about this in various contexts. It is the metadata that your browser sends...

Leo: Oh, that old word "metadata." That old black magic got me in its spell.

Steve: It's all of the headers, see, because in the headers, for example, are things like all of the different versions of add-ons that you've got. Like Java adds itself to the headers, and all these different packages have many-digit version numbers. And so you add them all, when you look at them as an aggregate, it ends up being that - and it was Panopticlick was the site...

Leo: Right, right, right.

Steve: ...that deals with this. It turns out you can still get a pretty good lock on a person. But all we have to do is rearrange the headers and scramble the data a bit and change it. So it'll be very easy for people to do, for example, browser add-ons which completely blow fingerprinting. Fingerprinting is only useful at the moment because all that data is relatively static. But none of it needs to be kept static.

Leo: Right.

Steve: So anyway, we just got yesterday Firefox v22. It's fixed 14 vulnerabilities. And that's why I was saying I call this the countdown, or the count-up, to 23. We're one version away from third-party cookies being blocked by default in Firefox, which is still my go-to browser. Chrome has just gotten so bloated, Leo. I launch it, and I watch my memory just collapse. So I'm hoping at some point that Google will come back to that.

And remember we talked about it here, how much focus the Mozilla folks put on memory. They got to a point where they said, okay, we've just got to stop and fix our memory consumption problem. They had leaks, and they had memory that was just, I mean, it's sort of natural in the development cycle to be adding features, and under deadline, and you're under the gun. And so you're writing code, and it works, and then you just send it. Well, it all takes up space.

And so the notion of examining memory consumption is very similar to examining security problems, that is, authors who are just writing code to make it work aren't always thinking about security. They're not always thinking about memory consumption because they're just like, well, okay, I need this much memory, and so they grab it. Or if they're not sure how much, they grab more than they need because, if you grab less than you need, then you're in trouble. So it's a sort of a different phase of, in the same way that you audit an app for security, you can audit it for memory consumption.

And we'll remember that it was about a year ago that Firefox really got serious about memory problems because it was out of control. And, frankly, that's where Chrome is now. I don't launch Chrome unless I really have to because it's just - it's just ridiculous. Opera is like a fraction of the consumption. And, you know, they all sort of work in the same way.

So with v22, just released yesterday, of Firefox, we got for the first time in Firefox a set of new standards, which Chrome already has, called WebRTC. And I've not yet done a deep dive into it. So I'll just quote from TechCrunch that explained it well. TechCrunch said: "WebRTC allows developers to create web pages with built-in video and audio calls, as well as filesharing, without the need for any plugins or third-party software."

Leo: Yeah. This is something Google's been pushing and we want to use because, you know, we use Skype right now, but...

Steve: Yes.

Leo: ...we are very interested in having a WebRTC implementation that would just mean we could give you a private website, you'd go to it, and you'd be on.

Steve: So think about this. I mean, this is huge. This means that audio, video, and file interaction would be - will be, it is now in Chrome, it will be in Firefox - natively available so that just JavaScript, using new APIs to access the WebRTC, allows these apps to run in the browser. So, I mean, this is a big step forward towards the browser is everything model.

And so continuing what TechCrunch said, they said: "Until now, only Google's Chrome supported the budding standard in its mainstream browser releases. Now that Firefox [with release 22] also supports it in its stable branch, we will likely see a large number of startups and established companies examine this technology far closer. Microsoft so far remains the only major vendor who has decided to go ahead with a different" - oh, my god - "with a different standard for the same functionality. But I wouldn't be surprised if Internet Explorer, too, would support WebRTC out of the box in the near future." So that's cool.

And then the technology that is really interesting - not only because it has "asm" in its name, Leo. Yes, it is, they're calling it asm, a-s-m, asm.js. We've talked about this before. And it's really interesting. It is a - the guys working on JavaScript speed realized that a lot of time was being spent in JavaScript on things, on features in JavaScript that, eh, you could live without. Because JavaScript is sort of an automatic language, the way it creates and destroys variables, there's the need to do garbage collection. If it realizes that things you've referenced are no longer useful or no longer being referenced, and not going to be, then it says, oh, I can free up this memory.

Well, that's - there's like an overhead associated with being that smart which adds a layer of complexity to the entire language. So what they realized was, you know, if we defined, carefully defined a subset just of JavaScript, still JavaScript, but, like, only allowed certain features and explicitly disallowed a bunch of these fancy ones, we could make this much faster.

Leo: Yeah.

Steve: And so that's what asm.js is. It is a subset of JavaScript that screams. And there is, for example, a compiler called Emscripten which can compile C and C++ code into this subset of JavaScript. And compared to native performance, it's running as fast as only half as fast, which is a bad way of saying it.

Leo: You only have a 50% hit.

Steve: Yes. Which is amazing because, I mean, native code is screaming on today's processors. And so now we're talking about completely browser, I mean, we're talking about a web page being able to execute code that is only half as fast, which is still amazingly fast. And in fact Mozilla has a page - shoot, I can't remember the name of it now.

Leo: I'm going through a presentation about it. And I guess the point being is that hand optimized - there's so many good optimizations that people don't use when they hand - when they write in JavaScript. And so hand-optimized code is not going to be as good unless you really pay attention to it.

Steve: Right. And so but the point is that, by formally denying access to the whole JavaScript language...

Leo: Right, a subset, yeah.

Steve: Yes. By formally denying access to all of the things that are expensive and slow, you end up with a subset fully useful which is extensive enough that you can write a compiler to compile standard C and C++ into asm.js, and it runs like a bat out of hell. So that's how I should have said it. Runs like a bat out of hell.

Leo: Bat out of hell.

Steve: Bat out of hell.

Leo: And I love it that it's C. I mean, I feel very comfortable with that. JavaScript always looked a lot like C. But...

Steve: Yeah, well, it's sort of - we've come up with sort of this agreed-upon pseudocode. If you look at articles in Wikipedia or in computer textbooks, they sort of use code that anybody can read because it's sort of like Basic, it's sort of like C, it's sort of like Pascal, it's just sort of this goop that sort of like, oh, looks generic. And that's sort of what JavaScript looks like.

Leo: Just running it through something like this also, as they point out, because it's got formal type checking, you're going to avoid a lot of common JavaScript bugs.

Steve: Yes, yes. Exactly.

Leo: Yeah. So this is good.

Steve: Yeah. So it's really nice.

Leo: It's not the first time somebody's done something like this. There's quite a few of these.

Steve: Yes. There'll be many different types. Like there's JIT (Just In Time) compilers and different approaches.

Leo: Well, Google has GWT, which is used in a lot of Android stuff, GWT, the Google Toolkit. Yeah, there's Coffee, there's, boy, there's a lot of stuff, CoffeeScript. Huh. Cool. So you think this will be widely adopted.

Steve: Okay. Well, it needs to go multibrowser. So at this point...

Leo: You just like it because it's asm [laughing]. If only somebody would write an assembly language JavaScript compiler. That's what we need.

Steve: Well, and Google does have their own native code project that they're working on. So we're seeing everyone wanting to move toward more performance and to give browsers enough speed that we can implement full applications in the browser and continue to add more power. So this is just another step in that direction. I'm glad to see the Mozilla folks doing it. I'm glad for what Google's doing. Ultimately we'll come up with a standard. And maybe Microsoft will support it. We can hope.

Bruce Schneier is actually where I picked up on an article in The New York Times that had one of the most interesting quotes from the EFF about all of this I have seen. But I'll hold that for a minute because what Bruce had to say, and his blog posting that caught my eye, was "New Details on Skype Eavesdropping." And you probably ran across this, too, in the last week, this "Project Chess," Leo?

Leo: Yeah.

Steve: Okay. So The New York Times article, this is Bruce writing: "This article on the cozy relationship between the commercial personal-data industry and the intelligence industry has new information on the security of Skype."

So now switching to The New York Times article, quoting from that: "Skype, the Internet-based calling service, began its own secret program, Project Chess, to explore the legal and technical issues in making Skype calls readily available to intelligence agencies and law enforcement officials, according to..."

Leo: This is way pre-Microsoft. This is under eBay.

Steve: Yes. Years, years. Yes, yes, "...according to people briefed on the program who asked not to be named to avoid trouble with the intelligence agencies. Project Chess, which has never been previously disclosed, was small, limited to fewer than a dozen people inside Skype, and was developed as the company had sometimes contentious talks with the government over legal issues, said one of the people briefed on the project. The project began about five years ago, before most of the company was sold by its parent, eBay, to outside investors in 2009. Microsoft acquired Skype in an $8.5 billion deal that was completed in October 2011.

"A Skype executive denied last year in a blog post that recent changes in the way Skype operated were made at the behest of Microsoft to make snooping easier for law enforcement. It appears, however, that Skype figured out how to cooperate with the intelligence community before Microsoft took over the company, according to documents leaked by [none other than] Edward J. Snowden, a former contractor for the NSA. One of the documents about the PRISM program made public by Mr. Snowden says Skype joined PRISM on February 6, 2011." So about six months prior to Microsoft's closing their acquisition deal.

So back to Bruce Schneier, who continues his blog, saying, "Reread that Skype denial from last July, knowing that at the time the company knew that they were giving the NSA access to customer communications. Notice how it is precisely worded to be technically accurate, yet leave the reader with the wrong conclusion." And this is Bruce speaking, saying, "This is where we are with all the tech companies right now. We can't trust their denials, just as we can't trust the NSA - or the FBI - when it denies programs, capabilities, or practices. Back in January, we wondered whom Skype lets spy on their users. Now we know."

Leo: He points out, he says, Bruce says, you can't trust the NSA, and you can't trust these companies. Their denials are meaningless. So we just don't know what's going on.

Steve: We just, yeah, we don't know. In that article, as I mentioned before, was a quote from Dan Auerbach, who's a technology analyst with the EFF, the Electronic Frontier Foundation. And I thought this was the best thing I read. He said, "We reached a tipping point, where the value of having user data rose beyond the cost of storing it. Now we have an incentive to keep it forever." And I think that's just exactly it. We know what's happened to the cost of storage in the last few years. It's just - it's ridiculous how large drives have become and how inexpensive per byte storage has become. At some point the cost to store drops so low that the perceived value of keeping everything outweighs it. And as he says, we've crossed that tipping point.

Leo: Oh, yeah. Oh, yeah.

Steve: Thus five zettabytes in Utah, five billion terabytes.

Leo: I paid 250 bucks for this thing called a Memoto. It's a camera that records an image every 30 seconds. You wear it around your neck or on your lapel. And then it uploads. It does a lot of parsing of this data, puts together stuff...

Steve: Life blogging?

Leo: Yeah, it has GPS in it, uploads it to their server, stores it. I mean, I think this is a good - and they say it's great for later when you get Alzheimer's because you can just go back and look.

Steve: "I never went there." "Oh, yes, you did, Grandpa."

Leo: And the name recognition, you know, there's name recognition, and Picasa does it; Facebook does it. So you just kind of apply some of these engines to it. Pretty soon you've got a whole record of everything you ever did.

Steve: And so does the NSA. Because they're happy you're doing this, Leo.

Leo: I couldn't care less if they do.

Steve: I know.

Leo: You know? Go ahead, fine. If you're really that - that's why, well, the trick is to overload them. Say you have so much on me, you've got - what are you going to do? I know they're - I'm using PGP. I'm dead already.

Steve: We ought to just have - create an app that just sends out pseudorandom noise. Because that looks like encryption.

Leo: And then they're going to record it all. But that's the, you know...

Steve: Exactly [laughing].

Leo: Yeah. If there were somewhere...

Steve: Just spew it out of our ports.

Leo: If there were somewhere one could go, I would consider going there. But I don't know - there's nowhere. You know, everybody's...

Steve: The only thing that I think is kind of cool is when we have an app, when we have like an instant messaging app, where we absolutely know that it is secure, like the one I talked about last week, Threema, where you actually, to get the highest level of security, you have to face your phones, they have to face each other, and they each look at the other's private key. I mean, at the other's, like, Q - what is it, QRC? I can't think of the name.

Leo: What?

Steve: You know, the little square barcode, Q something.

Leo: Oh, QR code, yeah.

Steve: QR code. Yeah, QR code. And that you have - they have to be physically in proximity. Then they can exchange directly their public key that matches their private key, and then you get three green dots. And forever on after that, you know that, when you send a text through the Threema system to a recipient...

Leo: I like this.

Steve: I do, too. And it's like...

Leo: You have to meet them in meatspace at some point; right?

Steve: Yes, exactly.

Leo: For this to make sense. But it's - that's...

Steve: Yeah, you have - yes. You have - there's three levels of security. You can use their distribution server, but then you only get two dots, and I think it's orange, and one dot is red. And it's only when you have, as you said, meatspace, m-e-a-t, that you have proven you've been in proximity to the other device. Now you have absolute authentication.

Leo: I love this.

Steve: And afterwards, I just - I like the idea. I just - I guess it's a little thumb of the nose. I'm sending this, and absolutely nobody else on the planet can possibly intercept it and read it. I don't need that. No, I don't need it because, you know, I'm talking to Jenny about when we're going to meet for dinner. But it's just like, yeah.

Leo: Well, that's my plan. I just want to swamp them. I think if all of us honest folks just swamp them, I mean, IBM estimated that it's 55...

Steve: It'd be good for the Utah real estate market, Leo.

Leo: I think we can swamp - I know zettabytes are a lot. But already there's a lot - what was - IBM had a Big Data, total amount of data generated every day. What was it? It was...

Steve: Well, this whole move to going digital, it's, I mean, we've all seen the quotes. It's like the total amount of data that exists that is created every week is, like, more than ever existed in the history of man or something like that. I mean, it's just - it's gone exponential already.

Leo: So according to IBM, and this is on their Big Data page, every day we create 2.5 quintillion bytes of data. 90% of the data in the world today has been created in the last two years alone. So I think we can - I don't know, how many quint - not 2.5 quintillion bytes. How many zetta - how long before we fill up a zettabyte?

Somebody do the math. And all we have to do, probably all we have to do is double or triple that, and there'll be so much noise, let them try to find the needle in a haystack.

Steve: Yup.

Leo: That's why I'm going to record every 30 seconds a high-res picture of what I'm up to.

Steve: [Laughing] And send it up.

Leo: And send it up to the cloud. Go ahead. Enjoy, NSA.

Steve: Send it on.

Leo: It's all yours, baby.

Steve: Now, the other little bit of interesting Edward Snowden news to arrive was picked up by The Daily Beast, who interviewed Glenn Greenwald, who's our friend at the Guardian who's been sort of the main contact person for Snowden.

Leo: Poor Glenn Greenwald is getting trashed.

Steve: I know. And it's...

Leo: He's a journalist, folks. You know?

Steve: Yes.

Leo: This is what we're supposed to do.

Steve: Boy, I was disappointed that...

Leo: First Amendment. David Gregory, what a dick.

Steve: Yes, I was just going to say, David Gregory's question on Sunday was just - it was ridiculous. Although he said, "To the extent that..." is the way he started it. And it's like, well...

Leo: It's like saying, no offense, but are you a spy for the bad guys? No offense.

Steve: Yeah, it was really, really, really...

Leo: He basically implied that, in his journalistic - that Glenn wasn't a journalist, or that in his journalistic endeavor...

Steve: He was aiding and abetting.

Leo: He was aiding and abetting. And, you know, this is called the First Amendment. This is called, you know, I just - it's shocking.

Steve: Yeah, I - yeah.

Leo: I'm telling you, if there were somewhere I could move, I would. But I don't - there's nowhere to go.

Steve: We're in the best place.

Leo: Yeah, we are. At least we can talk about it freely still. Nobody's pounding on our door. No storm troopers.

Steve: And that's why I love the country. I mean, I do.

Leo: It's not over yet. Let's fight for it.

Steve: These are - and of course we know that the Supreme Court did the right thing, I think.

Leo: They did one bad thing last week, or Monday, and one good thing today, so.

Steve: Yeah. Anyway...

Leo: They're 50-50.

Steve: Anyway, this is not a surprise, but this was interesting to have it made more explicit. And this is Eli Lake, writing for The Daily Beast, who interviewed Glenn Greenwald. Eli wrote, "As the U.S. government presses Moscow to extradite former National Security Agency contractor Edward Snowden, America's most wanted leaker has a Plan B. The former NSA systems administrator has already given encoded files containing an archive of the secrets he lifted from his old employer to several people. If anything happens to Snowden, the files will unlock.

"Glenn Greenwald, the Guardian journalist who Snowden first contacted in February, told The Daily Beast on Tuesday" - which is yesterday from this podcast's recording date - "that Snowden 'has taken extreme precautions to make sure many different people around the world have these archives, to insure the stories will inevitably be published.' Greenwald added that the people in possession of these files 'cannot access them yet because they are highly encrypted, and they do not have the passwords. But,' Greenwald said, 'if anything happens at all to Edward Snowden, he told me he has arranged for them to get access to the full archives.'

"The fact" - now back to Eli. "The fact that Snowden has made digital copies of the documents he accessed while working at the NSA poses a new challenge to the U.S. intelligence community that has scrambled in recent days to recover them and assess the full damage of the breach. Even if U.S. authorities catch up with Snowden and the four classified laptops the Guardian reported he brought with him to Hong Kong, the secrets Snowden hopes to expose will still likely be published.

"A former U.S. counterintelligence officer following the Snowden saga closely said his contacts inside the U.S. intelligence community think 'Snowden has been planning this for years and has stashed the files all over the Internet.' This source added, 'At this point there is very little anyone can do about this.'

"The arrangement to entrust encrypted archives of his files with others also sheds light on a cryptic statement Snowden made on June 17 during a live chat with The Guardian. In the online session he (Snowden) said, 'All I can say right now is the U.S. government is not going to be able to cover this up by jailing or murdering me. Truth is coming, and it cannot be stopped.'

"Last week NSA Director Keith Alexander told the House Permanent Select Committee on Intelligence that Snowden was able to access files inside the NSA by fabricating digital keys that gave him access to areas he was not allowed to visit..."

Leo: Wow.

Steve: I know, "...as a low-level contractor and systems administrator. One of those areas included a site he visited during his training that Alexander later told reporters contained one of the Foreign Intelligence Surveillance Act (FISA) court orders published by The Guardian and The Washington Post earlier this month."

So clearly our intelligence agencies are really looking closely at what's come out that they're aware that Snowden has, and then backtracking all of their logs of his access to figure out what more he may have. But anyway, so what's interesting is that there is a dead man's switch. He's created that. And it's funny, Leo, because there's so many times now in, like, plots in movies and on TV, where it's like, well, okay, why didn't you send - put this all in an envelope and send it to your attorney, or send it to your mother, or do something, you know...

Leo: Well, to be fair, a lot of movies they do that also. I mean...

Steve: Yes, yes. Yeah, and so you're right, it's been a - it's a plot device. And but I don't put it at, I mean, it's funny because we've seen criticism of Edward a lot in the last couple weeks. But I watched the video a couple times of him being interviewed. And it seems very clear to me that this guy is no dummy and that he did all of this on his own schedule. Everything has been on his own schedule. And even his move to Hong Kong that was roundly criticized by all the talking heads that wanted to be critical...

Leo: Paid off.

Steve: That all worked exactly according to plan also. And now he's got Julian Assange and his team working with him, and he's now moved to Russia, and asylum in Ecuador is apparently in the works. So anyway, we'll see how this plays out one way or the other. We'll certainly talk about it from a news angle.

But given that this was all on his schedule, I wouldn't be the least bit surprised if this is exactly true, that somehow, something needs to - we've even talked about, what was it, Daniel Suarez's network-aware technology. I mean, it would be easy for bots somewhere to be monitoring news and reading news stories and act at that level. Or just he has to do something on the Internet, send email to somewhere, and it bounces around or who knows what every so often in order to keep something alive. Or maybe it's just an agreement with people who know that they have this information. If it appears that I disappear, press this button, or do this, or send this to somewhere, whatever.

Leo: Didn't the same article say that Greenwald's computer was stolen after he mentioned on Skype that he had data on this computer?

Steve: Ooh. I hadn't...

Leo: Yeah. He's in Rio. I mean, this whole thing is...

Steve: Wow.

Leo: If, as some say, Snowden is making it all up, it sure is annoying somebody in D.C. a lot. A lot.

Steve: Yeah. Yeah, well, and in fact, in that first video he made, he said, "Yeah, we've got the CIA field office just down the block from here," he said. "I imagine they're pretty busy right around now." Yeah, I bet you do, and they are. So anyway, enough on that.

There was a bunch of - I sent something out on Twitter that I tweeted. I tweeted: "To the NSA's question asking us, 'If you have nothing to hide, why is your communications encrypted,' I ask in turn, how would you know?" And I got back a bunch of fun tweets. But the best one of all came from Nathan Long, tweeting as @sleeplessgeek. And he said, "One might say I have nothing to hide from people I trust."

Leo: Oh, there you go.

Steve: And I thought that was exactly right.

Leo: That's a good way to put it. That's a good way to put it.

Steve: That's exactly - that's exactly it. It's not that I have anything to hide. I have nothing to hide from people I trust.

Leo: Right. But we just can't trust the government to not misuse that information.

Steve: Well, for me, it's the gloves are off when the Director of National Intelligence...

Leo: Lies.

Steve: ...flatly lies to Congress. That's it. It's like, sorry, that's - you can't - our system depends upon oversight. And it was in the law. It was built in. And if you short-circuit it, then, sorry, you don't get anything anymore.

Leo: Yup.

Steve: A couple TV notes: Monday night we had the premiere of the Stephen King novel-based TV series "Under the Dome."

Leo: Oh. Did you read that? Did you like it?

Steve: Well, no. However, I heard...

Leo: Thurrott liked it. He didn't like the end, he said, but he liked the book.

Steve: Yeah, I heard that. Actually someone - I tweeted earlier that the first episode, the premiere episode was Monday. It is being repeated, I didn't note what night. But probably, like, this weekend.

Leo: Eh, it's on demand. What do you need to...

Steve: Okay. So for all of our listeners, for what it's worth...

Leo: Get Hulu Plus. Watch it on Hulu Plus.

Steve: This is not giving anything away because you know this from even the name and the previews. But a small town suddenly gets instantly cut off by a force field which just appears out of nowhere, and it's the story of that, and based on a Stephen King novel. Now, I downloaded it because I thought, okay, I can't wait for, like, this to be doled out to me a week at a time. So I thought, I'm just going to read the book. Oh, my lord, is it big. I mean, even the Table of Contents...

Leo: I know.

Steve: Even the Table of Contents scrolls on and on and on.

Leo: I'm on a break from Stephen King. I finished "The Stand," and now I need a breather for a year or two, and then I'll look at "Under the Dome." He writes a lot of words.

Steve: Boy. So the response I got, that I saw before I shut down my Twitter client for the podcast, was that the people in the series and both inside and - the people in the book, both inside and outside the dome, don't behave in a realistic manner. And if that's true, that would really annoy me because I hate when writers do that, when writers make their characters do things that are, like, dumb. It's like...

Leo: I think that's a fair knock on a lot of King's stuff.

Steve: Oh, well...

Leo: Yeah. Have you read any of his other books?

Steve: I've never been a Stephen King reader because I just - I've got so much sci-fi going on.

Leo: He's a great writer. And I think...

Steve: I like the movies that have been made from his stuff.

Leo: Yeah. He's a very good writer. But I think you could - that is a fair complaint to make about almost everything. It's so plot-driven that the characters have to work in support of the plot.

Steve: Yeah.

Leo: And this isn't always, you know, "The Stand" had a number of characters that just were wooden because they were forwarding the plot. They weren't...

Steve: M-O-O-N.

Leo: Yeah, there you go. You did read "The Stand."

Steve: Oh, I saw it.

Leo: Oh, it was a show, okay.

Steve: Yeah, yeah.

Leo: Yeah, that's a good example, actually.

Steve: Yeah. So also "Falling Skies," I mentioned it before, I wanted to reiterate, wow. Season 3 is really, I mean, it's - I'd be hard-pressed to say whether it's worth putting up with the first season and most - the first half of the second season. The second season began to get really good. They're really onto something now. It is, I think, the best of the low-budget sci-fi TV series that are on right now is "Falling Skies." I think it's great.

Leo: It's low-budget?

Steve: Well, yeah, I mean, they're all trying to not spend as much money as they, I mean, to minimize the cost. And so special effects are like, eh, okay, I mean, they're fine. But they're character-driven sci-fi settings.

Leo: I kind of like character-driven. I don't mind character-driven.

Steve: Yeah. And it's gotten - it's really gotten better. And I got two tweets, I will just say, in response to my recommending "The Killing," the series on AMC, which I talked about last week.

Leo: Oh, whoo, I can't watch that [vocal shuddering].

Steve: And one, @Abdullah_Hamad in the UAE, he tweeted: "Thanks for recommending the TV series 'The Killing.'" He said, "The acting is beyond excellent. They all deserve an Oscar." And that's what it is, Leo. I just - I want to, like, watch it again just to watch the acting. It is superb. And then someone, @md_seuss tweeted, he said: "Thanks, I think, for the recommendation on 'The Killing.' I can't stop watching it. It is great."

Leo: Good.

Steve: So he's as sucked in as I was, which is cool.

Leo: Do you want to do a SpinRite testimonial?

Steve: Well...

Leo: In some ways this whole show is about SpinRite, so...

Steve: Precisely.

Leo: Yeah. So we'll do that in a second. We'll talk about what you've learned, some Intel history, useful and interesting.

Steve: That I needed. Something that I needed from the 20 years past.

Leo: Wow.

Steve: Yeah.

Leo: Let's talk about memory. Steve Gibson, Leo Laporte. Time to learn some history that is still of great value.

Steve: So, okay. A little to set the stage here. This was driven by my current R&D effort, essentially, for the next release of SpinRite, which I'm calling 6.1. My intent is not to rewrite SpinRite. That would take a long time. And it's really not necessary for what SpinRite is today. I do, because we've seen that SpinRite is able to recover SSD drives - like as far as we know it's got a great track record of doing that. That to me says, okay, it's not going away anytime soon.

So I'm fully looking at a v7 which will actually be a restart because I want to add features that the current user interface just can't handle and, in fact, that whole architecture is not designed for, like going into the file system, having it be file system aware, allowing you to say I want to pull this file off of the drive, rather than just fix the whole drive. Or I want to pull all of my documents. Or I want to prioritize recovering files over recovering space. Or I want to clone a dying drive to another drive. These are things that people have asked about.

And it's like, well, yeah, but that's not what SpinRite - that's not the way it was originally designed. It was first designed when people only had one hard drive. It was, like, the most expensive component of their entire computing system because 10MB was so expensive back then. And so the goal was just to fix that one huge investment you had.

People just weren't plugging hard drives in right and left. So in terms of my overarching plan, I'm fully intending to scrap everything I have right now. But not yet.

So what 6.1, and there's probably going to be a .2 and .3, the focus is fix everything that I can, which in fact brings SpinRite current for the features that we've talked about, that all these testimonials are based on and so forth, but just makes it better. But not different. Different will be v7. And I'm going to have to go away for a long time to create that. So we need an update to v6 in the meantime.

So I'm very methodical in the way I approach things. The first thing that I looked at was that SpinRite sometimes has a problem just with FreeDOS booting. And SpinRite, before SpinRite even gets a chance to run, some users have FreeDOS explode on them, and they never get a chance to run SpinRite. And the good news is that they'll shoot an email to, I mean, it doesn't happen often, just it's a low, low, low level. But it's been on my radar for a while.

Greg is always able to fix that, just by helping them use MS-DOS rather than FreeDOS. We include FreeDOS because it's license-free. MS-DOS you have to have Windows. But there are sites on the Internet where you can download it. But even in XP, that doesn't have DOS, you can have XP create a DOS-bootable floppy, for example. And so there's - MS-DOS is in Windows. It's still there. It's just you get to it when you say I want to make a bootable floppy. So we can fix that.

But so I started this project a couple months ago. And I started looking at the FreeDOS kernel. Why is it blowing up? Well, it turns out that FreeDOS does something that MS-DOS doesn't, which is it tries to open a dummy file from every drive, on every partition of every drive it finds. And it does that to populate some tables that are used to manage the FAT systems on each of the drive's partitions. And it does that, it turns out, because some random Norton utility blew up if you ran this Norton utility from the Dark Ages on FreeDOS and asked it to access a drive that FreeDOS had not accessed before. And so they added this thing to FreeDOS where it went out and tried to open a file on every partition of every drive. And in some cases of a particular type of disk damage, that will cause FreeDOS to die.

So the first thing I did was create a custom kernel. So we have that now. I added a new line to config.sys. It just says "skipinit=1." And so in the FreeDOS kernel that will be coming with the next version of SpinRite, in the config.sys it'll say "skipinit=1" because SpinRite doesn't need that. And actually it's dangerous, as we've learned, to have FreeDOS do that. So, okay, first problem was solved.

Second problem was the question of Mac compatibility. And I talked about how I rewrote the keyboard interface, and now it's running on the Mac. It's running on my MacBook Air without any problem. There's more work to do there over on the booting side. But essentially that's resolved.

So the next problem, and this was substantial, was to get SpinRite to operate all drives in what's called UDMA, Ultra DMA mode. Many BIOSes do that, but not all do. And so if you're - and SpinRite today has still been using the BIOS to perform its bulk data transfer. That is, all of the reads and writes that it does, it just asks the BIOS for. The logic behind that is we just get complete compatibility because the BIOS always knows how to talk to the controllers that are on the motherboard.

But the BIOS doesn't always know how to do it fast. And it really doesn't need to because in today's world the BIOS just gets the OS booted, and then the operating system uses its own drivers in order to talk to the hardware and take over. Which is why anyone who's ever set up a new operating system on a PC-style machine knows that the motherboard comes with drivers, and you quickly install those in the OS so that it's able to talk to the hardware efficiently. So essentially SpinRite needs to add to itself, it needs to acquire an understanding of all of the latest hardware, the mass storage hardware that exists on motherboards.

The other thing that it needs to do - oh, and I should mention, so I now have so-called "PCI bus enumeration." That's all done and working. I wrote it in Windows. The gang in the grc.spinrite.dev newsgroup pounded on it and tested it. And so now SpinRite has the ability to understand the PCI bus and fully explore every controller on the PCI bus, finding all the mass storage controllers that are accessible in the machine. So even if the BIOS doesn't support something, it's still a PCI peripheral, and SpinRite will now find it.

One of the problems that we've had with performance, because performance is one of the areas I'm really focused on, you know, people have talked about how SpinRite can spend weeks recovering, like struggling on a really damaged hard drive. Typically it takes maybe hours. But it can take weeks. Clearly, a huge aspect of convenience would be speed. We all - none of us want SpinRite to be slow. So one aspect is that SpinRite will incorporate its own low-level device drivers to talk directly to the hardware on the motherboard. And in fact one of the things that has happened with drive evolution and controllers, we had the original IDE interface, also known as the ATA interface. Then when PCI came along, there was something known as "native PCI" which was sort of a bridge between the way the IDE controller worked and the PCI bus.

And then the latest type of controller is called an AHCI. This is an Intel standard, Advanced Host Controller Interface. AHCI mode is believed to be faster, but actually isn't faster, than native, the so-called "native IDE" or "native PCI" mode. And BIOSes differ in the way they are configured. Most people just, if they're building a new system from scratch, they will install the OS on their BIOS or on their motherboard and run it. Most BIOSes are still defaulting to this compatibility mode, this native PCI which is compatible with IDE, rather than using AHCI.

People believe AHCI is faster. Arguably, it's faster in a server setting because it allows drives to be given many things to do at once and for the drive to schedule their completion and essentially, whether it's reading or writing data, to perform that as they are able to in order to improve the overall throughput through the drive. But that really requires that the system have many different sorts of things to do at once. And typical single-user workstations don't. The system sits around most of the time, waiting for something to do. So it really isn't clear that that's a benefit.

One of the requirements that SpinRite has imposed, SpinRite 6 has imposed on its users is that, if SpinRite doesn't see the drive because the motherboard has been set for AHCI mode, that the user who wants to run SpinRite manually change the BIOS over to this compatibility mode. And then SpinRite will see it. You run SpinRite, it does its recovery job, and then you switch it back. The good news is all of that will be automatic because it turns out that all of the AHCI controllers also support the original compatibility mode, and SpinRite will be able, without you having to mess with the BIOS, to just make the switch for you.

But the big problem is buffer size. The traditional track on a hard drive was 17 sectors. An MFM, a Modified Frequency Modulation track had 17 sectors. Then we went to RLL, and we got 26 sectors. That's a 50% density improvement with run length limited encoding. We got 26 sectors. But in the BIOS there are only 5 bits to specify the sector number. Well, 5 bits gives you from zero to 63 in binary. And so that's technically 64 different possibilities, except that sector zero was never valid. There was never a sector zero. For some reason, we don't know why, there's a head zero, there's a cylinder zero, but sectors were always numbered from one. So who knows why. So that means you could only have 63 sectors on a track.

Well, that 63 sectors, if we round it up to make the math easy, that's 64. And sectors are 512 bytes each, which is to say half a K each. So that means that a track buffer was 32K. The problem is that SpinRite 6 today is working with 63-sector buffers, that is, slightly smaller than a 62K buffer, and transferring chunks of the hard drive at this buffer size back and forth as fast as it can. What we really need, though, are much larger buffers.

And this is where I hit a problem last week, essentially, as I was methodically moving forward, solving one problem after another in this update of SpinRite's low-level data transfer architecture. I needed megabytes of buffers. In fact, you know me, if I'm going to do it, I'm going to do it all the way. All of the late-model drives, and actually drives that have followed the ATAPI standard, or the ATA standard, for many, many years, they support a transfer of up to 64K sectors. So that 64K, 512-byte sectors, you can actually tell the drive, I want you to transfer that block. Well, that is 32MB. And it turns out that's 32 binary megs, which is 33.55 actual million bytes of data in a single transfer.

Well, now, that's fabulous because what that means is that the drive would start at whatever sector we tell it, and it would either write data or read data nonstop until it has transferred 32MB of data from a single request. And the one thing that drives have gotten right - remember SpinRite was born to fix the interleave of drives, in order to allow them to transfer as much data as possible per rev.

Leo: That was what SpinRite was designed for originally?

Steve: Yes.

Leo: How interesting.

Steve: Yes.

Leo: I probably used it that way. I remember changing the interleave on my old Seagate...

Steve: Oh, exactly, the ST-253, I think it was.

Leo: 253? Yeah.

Steve: 255 maybe?

Leo: One, I thought. But anyway...

Steve: 251, the ST-251, you're right.

Leo: Yeah. It was the first to not use - what's the successor to MFM it used anyway?

Steve: It was RLL.

Leo: RLL, that's right.

Steve: It was the early RLL drive. And they were all interleaved wrong.

Leo: Yeah.

Steve: And so SpinRite was - I wrote it to be an interleave optimizer.

Leo: So I bought it in v1.

Steve: Yeah.

Leo: Wow.

Steve: And the reason I had to add all this other technology is that sectors were marked bad in logical sectors. But if I rearranged the interleave...

Leo: Oh, yeah.

Steve: ...then the...

Leo: It would be meaningless.

Steve: ...the error in the sector went to a different logical sector because it was in the same physical location. So that meant I had to detect errors. I had to do pattern testing. I had to verify the surface. I had to do all this other stuff just to interleave, just to safely re-interleave the drive.

Leo: How funny. How funny.

Steve: And so of course we lost that. No one is doing messy with their interleaves anymore. They're all 1:1.

Leo: Right.

Steve: Yet because - oh, and the other thing is, if I was going to - because the only way to change the interleave is to do a new low-level format. I had to re-low-level format one track at the new interleave. Well, if I'm going to do that, then I'm going to wipe out all the data. That means I have to absolutely recover absolutely everything that is there. So all of the stuff that we use SpinRite for now is actually a side effect of the fact that I had to do - as you know, my nature, it was going to be right. And the only way to do it right was to absolutely do data recovery and then surface analysis in order to safely and properly re-interleave the drive.

Leo: And I had those 251s. I upgraded my FidoNet BBS to those RLL drives. And I remember we were trying to get the most - I was trying to get the most speed out of them. And I remember that that's where I met Steve for the first time. That's great.

Steve: And I did something in my InfoWorld column that was the most popular thing probably I ever did. It was called "Steve's Dream Machine." And...

Leo: Yes, I remember that, yes.

Steve: ...one of the things I noted was that the drives we were all using actually had some unused cylinders at the end. And if you did something, I don't remember the details now, to, like, fudge the size of the drive, you could then get two maximum-size 32MB partitions on the same drive. You could get a C and a D, and neither of them could possibly be any larger because of the sector count problem that we had in that version of DOS. And so everyone loved the idea that you could just sort of like push - you could get just a few more cylinders out of the drive and then get two beautiful 32MB partitions, a C and a D. So that was all craziness that we were up to back then.

But so here I am, with drives in the world, everyone listening to this has drives that can transfer 32MB at once. And of course, if I'm going to do this, I'm going to figure out how to make that happen. But the problem is SpinRite is a real mode program. DOS, FreeDOS, MS-DOS, all DOS is real mode. Real mode is what was originally - there was nothing called "real mode" in the beginning because there was no alternative mode. So no one called it "real mode" when it was the only mode you had. It was just the chip, the 8088 and the 8086. The 8088 was in the original IBM PC. And it allowed you to access 1MB of memory.

And remember we had the Apple II at that time. And that allowed you - it had a 16-bit address bus, and 16 bits allows you to do 64K. So it allowed you to do 64K of memory. And so one of the big deals about the IBM PC was that it had this next-generation Intel processor. There was the 8080, and it was very much like the 6502 chip that was in the Apple II, similarly had a 64K memory size limitation.

So for the 8088, Intel added 4 bits to the address bus. So the 8088 had a 20-bit, 16 plus 4, a 20-bit address bus. And that allowed it to access 16 times 64K of memory, which was a meg. And because that seemed like an insane amount of memory back then, they allocated, they just took up the upper third, 384K was just sort of squatted on by the display memory, and the BIOS was up in the high memory area, and ROMs that came with different controllers that you might plug in and so forth. Everything lived, all of that, sort of the I/O space for video and BIOS and things, was in the upper 384K. The lower 640K was RAM available to software. And of course, oh, 640K, who's ever going to fill that up.

So that was the 8088, with 20 bits of address space. And that is the world that SpinRite was born in, and it's the world that SpinRite still exists in. When you boot FreeDOS, you're in so-called "real mode." It wasn't until the next chip, the 8286, where Intel introduced something called "protected mode," and it was because they added protected mode that they said, okay, well, what are we going to call that other mode? Like what it's always been before? What the 8088 and the 8086 and so forth chips are? And they said, well, let's call that the "real mode" because that's all, for one thing, no software understood protected mode.

When they came out with the PC/AT, that was - it was the PC/AT that had a 286 chip in it. And they again added 4 bits, so that had a 24-bit address bus, so now you could get 16MB of memory, although it was incredibly expensive, and nobody did that initially. So that's why they were calling the original mode the so-called "real mode."

Okay. So here I am today with SpinRite. When all of these processors boot up, they are in real mode. The BIOS is in real mode. DOS is still around, it turns out, because it's still being used as a loader for other operating systems. And so real mode has never gone away. It's still supported in the very latest chips because it's still the foundation of how these systems operate. So SpinRite operates in real mode. And real mode knows, it has this memory concept that I was just describing for the original IBM PC of 1MB.

Now, that has never been a problem for SpinRite, ever, because the code itself, we often remark about how small it is. It's written by hand in assembly language. It was less than 64K for several generations. It was a COM file, which is just an image that gets loaded into memory. And then it had to outgrow that with SpinRite 3, I think it was. But it's always been able to operate with relatively small buffers.

For 6.1 I need to change that. If everyone's drives can transfer 32MB at a time, that's what I want SpinRite 6.1 to be able to do. But I'm in real mode. And there's no way to access more than a megabyte in real mode. That's always been the case. And protected mode is completely incompatible with real mode. I mean, they're oil and water. They're just - they just don't get along in any fashion. But of course, when Intel created the 8286, they realized, I mean, if Intel has ever been anything, it's backward compatible. And really that's been the success story of all of these companies. Microsoft is backward compatible. They always arrange for their new operating systems to run the oldest code you've ever seen, I mean, way, way back.

Intel has always adopted the same policy, and that is forward, as they move forward, they're going to retain backward compatibility. So the 8286 chip similarly booted, just as all of the chips today do, even the 8386 and our Pentiums and our Core i7, everything, they all boot up in real mode. Intel then created the ability to switch into this so-called "protected mode." And in protected mode you finally had this notion of a supervisory process, sort of like the operating system actually exists as an entity, rather than just sharing memory along with the various programs that are running, and it's sort of as an executive. And it's able to manage the programs that are running underneath it.

And the programs using the so-called protected mode, this protection is then being protected from each other and protected from having direct access to the system's hardware which could allow the programs too much freedom and power. So the programs are constrained. So the way, back on the Intel 8088, it was a 16-bit processor.

I mean, it was still a 16-bit machine. But remember it had a 20-bit address bus. That is, it was a 16-bit machine that could access 20 bits' worth of memory, 1MB of memory. How's that possible?

Well, what Intel did was they created the concept of segments. And that's, I mean, "segment" is a word that probably many people have heard before. That was an innovation where Intel said, okay, we have a 16-bit machine. But we somehow need to roam around within 20 bits. We need to access more than 64K, which is all we can access with 16 bits. So they took a second register, the so-called "segment register," and they shifted it over, they shifted it left 4 bits. And the value of that register would then be added to the 16-bit value that the instruction was able to offer in order to generate a total of a 20-bit address.

And so that's the so-called "segmentation" concept that Intel processors have always used. Back in the 8088 there was - so the idea would be the segment register would sort of specify which 64K region within the megabyte of memory of the 8088 you were able to access. And that's what I programmed in. And SpinRite today still has a lot of that logic because it runs in real mode.

When Intel went to the 286, they said, okay, we're going to change the way segmentation works. Rather than taking the value of the segment register and multiplying it by 16, which is what shifting it 4 bits to the left does, we're going to still have a 16-bit segment register. But it's going to refer to a table in memory. And the table in memory will specify the starting address of the segment, and it will specify the size of the segment, that is called the "limit" of the segment. And all those cool other properties that we want segments to have, like is it a code segment or a data segment? Can you execute in this or not? Is it readable? Is it writable? What is the privilege level? So by creating sort of this indirection table, Intel gave us protected mode.

So the segment, the value in the segment register, back in real mode it's just multiplied by 16 and then added to what's called the "offset," the original 16 bits. Instead, Intel said, okay, we're going to have that refer to a table that has much more exotic properties. It can be any offset in memory, so that'll be 32 bits. And the limit register, we're going to keep that at 16 bits. But we'll have a granularity bit where the values either specify bytes or 4K chunks. And I'm sorry, the limit register was 20 bits, and the base is 32 bits. And the granularity specifies either bytes or 4K chunks. And the beauty of that is 20 bits of the register in 4K chunks gives you 4GB, which is 32 bits. So that's where this notion of 4GB of address space came from, and this 32-bit base address in this table.

But now Intel had a problem with their design because, back in the 8088, as instructions were being executed, and they were referring to segments and offsets, the hardware to do that was trivial. You would take whatever was in the segment register. You could shift it left by 4. Actually, hardware can do that instantly. You just sort of wire the bits over 4 spaces. And then you add the offset into that segment to get the 20 bits. So, I mean, that's like instantaneous. That's no trouble at all.

But with the design of protected mode in the 286, remember that the segments refer to tables in memory. So that would mean, as you were executing instructions, and these instructions were referring to segments, the chip would have to be going out and fetching the entry in the table that the segment register referred to, getting all of that data, the base of the segment, that's 32 bits, the limit. And then it would have to take the base, add that to the offset, then check it against the limit to see whether you were exceeding the bounds of the segment. So it was like, wait a minute. This thing's going to be slower than the 8088.

Well, the answer from a computer science standpoint is pretty simple. You use a cache, which is exactly what Intel did in the 286. The idea is that the segment registers, they're not changing all the time. They're loaded with a value when the code wants to work within that segment. And normally it sits in there within a given segment for a while. And it may be referring to data in several other segments. But it's roaming around within the segment. The segment itself is not changing.

So the idea is any time the segment register is loaded, only when it's loaded does a reference have to be made to this table in main memory to get all of the data that that segment register is referring to. And in fact it's called a "selector" now. It's selecting a set of descriptors that are in this table, and the chip caches them. Essentially, it reads them once from the table into its hardware, into its internal hardware, so that then all references to memory in that segment can be instantly fixed up. They can be added. They can be limit checked. They can be checked for read and write and code and data and priority level and all those things that were added to create the segmented, the sort of this constrained in control architecture as Intel was moving forward. So that's how the 286 operated.

Now, there was a problem with the 286. And that is there was no way to take it out of protected mode back to real mode. At all. There was an instruction in real mode for switching to protected mode. But the engineers at Intel thought, hey, this is a better mode. Why would you ever want to go back? And it turns out everyone did because nothing ran in protected mode at the time. And the BIOS was in real mode. DOS was still in real mode. When you plugged controller cards in, they brought their own BIOS. There was a video BIOS that controller cards had. There was a disk BIOS that disk controller cards had. They were all set for real mode.

So nothing ran in protected mode. So it was a catastrophe that you were unable to switch the 286 back into real mode. And so, believe it or not, the IBM engineers who realized they had a problem, they said, well, only thing we can do is to reset the chip. And so they actually did that on the fly. The original OS/2 operating system that was the early operating system for the PC/AT, whenever it needed to use the BIOS or do any I/O with any of the peripherals, it would do this on-the-fly reset. It would literally reset the chip. The chip would come back up in real mode, as all Intel processors always do, and they would save the state of the chip just before the reset in some of the RAM of the clock controller.

Leo: Oh, wow.

Steve: The clock chip had some unused RAM.

Leo: What a hack.

Steve: It was a hack, it was an unbelievable hack. And so when the chip would come out of reset, it would go to the BIOS, and then it would check the data in the clock RAM to see if it looked reasonable. Did it match the pattern of, oh, crap, I was just in protected mode, and now this is what I have to load my registers with, and this is where I have to go. And so it was an incredible kludge.

When they did the 386, Intel fixed this. This was clearly a mistake. I mean, it was just a - it was bizarre that you could not switch the chip back to real mode. Everyone wanted to. So the 8386 does allow you. There's a bit in control register zero, it's the zero bit, that you turn on, and now you're in protected mode. And you turn it off, and now you're in real mode. It's just like, I mean, it couldn't be any simpler. I mean, they completely fixed it.

But how did Intel emulate real mode in their advanced processors? And this is the key. How did they emulate real mode? Real mode, as I've said, has these segments that are limited to 64K. I mean, that's all you can address with 16 bits. And in real mode on a 386, on an Intel Core i7, on any of these chips in our Macs and in our PCs, we've got 32-bit registers. You have AL is the low 8 bits. AH is the high 8 bits. AX is them together, making 16 bits, and so-called EAX is the Extended AX, and that's 32 bits. We've got all the registers that are 32 bits. But in real mode we only have the lower 16 valid. We are only able to access 64K in a segment, no matter how much memory the system may have.

And so here I am looking at SpinRite in real mode and realizing that I need 32MB buffers. Yet I'm in real mode, and I cannot have them. And I can't run in protected mode because DOS doesn't, and SpinRite doesn't, and nothing does except protected mode operating systems, but that's not what I have. And I can't rewrite everything. That'll be v7. But still, the reason I want these 32K buffers is that disks have gotten so dense that if I'm only able to issue 63-sector transfers in 32K buffers, not megs, 32K, then I ask for the data, and it's read. Then I ask for the next. But in the interval, I've lost the next sector. And so I have got to go all the way around again. It's like the drive is mis-interleaved.

I mean, it works. SpinRite today works. We're selling it. People are using it. It's doing data recovery. But it is not as fast as it could be. I don't know yet how much faster SpinRite will be with 32MB buffers. But my guess is at least, at least twice as fast. Because I'm probably missing a revolution for every small block of sectors I ask for, I've got to go all the way around again and get the next block, all the way around again and get the next block. So drives today are optimized for reading in a forward direction without ever missing a spin. So I would be able to read - so we're talking about going from 32K to 32MB. So I would be saving a thousand revolutions of the drive when I transfer 32MB. And so it's worth doing.

Well, the answer to this is what Intel did was they simulated real mode with their protected mode technology. They had all that fancy technology with base addresses and sector limits and so forth. So when the chip comes up out of reset, the microcode, the firmware in the chip, it loads those caches. Remember that in real mode there are no caches. There's no segment descriptor caching in all of that. That doesn't exist. All we've got is the simple math of multiplying the segment register by 16, that is, shifting it over 4, in order to give us a total of 20 bits of addressability.

So the firmware behind the scenes, it loads these caches, the segment descriptor caches, with 64K limits and allows full read and write, code or data, no protection at all, because there was - you could do anything you wanted to with the memory in an 8088, in a real mode processor. There are no constraints. And so the only thing then that real mode is doing is multiplying the segment register by 16 and adding it. Behind the scenes there is the limit registers and the privileges and things. But you never see them in real mode. Since the only way to take the 286 out of real mode was resetting it, real mode was always exactly like that.

But remember that I said the 386, Intel realized they'd made a mistake. Obviously people needed to drop back to real mode all the time. So they made it as simple as resetting a bit to drop back to real mode. What Intel forgot was to change the cache for its segmentation when they do that. It turns out that when you drop the 386 or any subsequent processor ever made back to real mode, and they can all do that, although the firmware in the chip when it boots, when the chip first comes up out of reset, the firmware loads those caches with real mode segments. When you're up and running in protected mode, and you've been loading descriptors from RAM and loading the caches and so forth, when you turn that bit off, the only thing that Intel does is they go back to multiplying the segment register by 16. They never again touch those descriptor caches.

So what that means is it is possible, and I did it yesterday, which is why I'm so excited about this, it is possible to switch into protected mode and to put into memory a 4GB limit, that is, all F's, 5 F's for 20 bits' worth of F in the limit register. The granularity is set to 4K, so that stretches all the way to 4GB. Set all the privileges to "free," read, write, code, data, anything you want to do, fully privileged. Then you load each of the segment registers. There's a code segment, data segment, extra segment, stack segment, and then there's an F segment and a G segment. You load them all with that descriptor, which completely sets them for 4GB of access. Then you simply drop back out of protected mode to real mode.

Now you are back in real mode, everything works. DOS works. The BIOS works. SpinRite runs. Everybody's happy. Yet there is no limit any longer on the size of the segments. Since we've got 32-bit registers in all of our 386 and subsequent processors, you can then put full 32-bit addresses into those registers, and they directly access physical memory.

So what this means is that SpinRite 6.1 will - oh, and this is not something I just discovered. This is well understood, well known. It's been known for - actually we're not sure how long it's been known for. IBM got a patent on this that they issued the patent in 1994. They called it an "artifact" of Intel's protected mode. They got, in '94, they submitted the patent. It was granted in '97. Oh, and it was '97, August 24th of 1997, two days ago and 16 years. Meaning that, since patents have a 17-year life, it's got one year left of the patent.

Except that people were using this in the early '90s. Wikipedia says that game developers who needed more space, this is sometimes called "unreal mode." There's real mode, and this is called "unreal mode." Or sometimes called "huge real mode," "big real mode." Because it never officially named, it's known by all kinds of things. I call it "extended real mode" because it gives me access to what has always been considered extended memory in the original real mode context. And so it looks like there was active use of this predating IBM's filing and being granted a patent. So the patent is probably not valid. Everybody uses it. DPMI, the DOS Protected Mode Interface managers use it. Himem.sys, emm386.sys, everybody has taken advantage of this. So it's probable that there's plenty of prior art here.

What I wanted, though, was not to use any of those memory managers because they don't operate the way I need them to. They copy blocks for you down from extended memory, down into conventional memory or back up. So they sort of shuffle things back and forth. You never really get access to it. I needed SpinRite to have full direct access to the entire physical memory, up to 4GB. I'll now then with 6.1 have 32MB buffers and be able to transfer massive blocks of data at a time. And, since I've got 32-bit registers, SpinRite will simply be able to reach right up out there into the stratosphere and do whatever it needs to with the user data because of this weird little fluke in the Intel design.

Oh, and by the way, they can never change it because everybody uses it. It's one of those things where it may have been an oversight because, when you come up for the first time, you're absolutely restricted to 64K. It's only when you go into protected mode, change the "shadow cache," as it's called, the shadow descriptor cache, and then drop out, that you're like, oh, look, I can now access all of memory, and I'm in real mode. So it's the best of both worlds.

Leo: Cool.

Steve: Yeah.

Leo: Does this change how you do anything?

Steve: Well, all of the PCI utilities, all of the disk controllers, they have access to this full memory. So SpinRite will operate with the disk controllers, set them up, and instruct them to transfer 32MB of memory at a time. I will probably use several of these buffers. So, for example, while one is being transferred, SpinRite is working with the other. For example, it does the bit inversion, where it inverts all the bits and then writes it and then reads it back and verifies it and then inverts them again and writes it and reads it back and verifies it. I can use, essentially, a double buffering scheme so that there's data being transferred in one buffer while I'm busy inverting and verifying the other buffer and so forth. So, yeah, there's lots of games that I can play having access, essentially, to the entire system's memory from within real mode.

Leo: Steve Gibson. It's an education to listen to you. And a trip down memory lane, I must say.

Steve: Yup.

Leo: What fun. Somebody said you should write a textbook. I don't know if there's a lot of use for this particular information anymore. But it's fascinating. I guess it's somewhat useful.

Steve: Well, it's funny, too, because...

Leo: If you're writing low-level disk drivers.

Steve: Even Intel's contemporary documentation does - they get this wrong.

Leo: Really.

Steve: They say, yes, they say as long as you don't change the segment register, the cache will not be invalidated. It turns out that's wrong. And I went looking through all kinds of source code, like I was looking at the FreeDOS source code, and no one quite understands it the way I've just explained it. I've explained it in exactly the way I have watched it operate and verified it. So, I mean, there's a lot of misunderstanding and confusion about exactly why this happened and how it works. As far as I know, what I've just explained is the whole story.

Leo: Hey, an update on the Snowden case. I'm reading The New York Times blogs. It's not what you think. When the people came to Edward's house, particularly lawyers advising him, he would have them put their cell phones in the fridge to block eavesdropping. Apparently...

Steve: That's pretty clever, though.

Leo: It's very smart. It's got metal walls.

Steve: The guy is not stupid, yeah.

Leo: So just a tip for anybody who wants to make your cell phone have a little privacy.

Steve: It's a kitchen-based Faraday cage.

Leo: Yeah. Yeah. You might want to put it in a plastic bag to keep it from getting wet, condensation. And be careful when you take it out. Wow.

Steve: It's interesting, too, because you could tell people to turn it off, but do they really?

Leo: Right.

Steve: But when you tell them, okay, we're going to stick it in here with the...

Leo: We're going to put it in the fridge.

Steve: We're going to put it with the lettuce and the onions, that's pretty clear.

Leo: A stainless steel martini shaker is the other [laughing]. It's a perfect Faraday cage, according to this article in The New York Times. So everybody get a stainless steel martini shaker and...

Steve: That would, that actually would be pretty good. I've always - I marvel at the fact that metal on metal somehow doesn't leak. Every time I see the bartender shaking this, I'm thinking, how is that not leaking?

Leo: I know, it should. You need a grommet, an O-ring or something.

Steve: I want a grommet, yeah.

Leo: Yeah [laughing]. Just love it. Steve, you are amazing. What a fun show this was. I hope everybody got a kick out of it and maybe even learned some useful information, information about Intel history. Steve is at GRC.com. That's where you'll find 16Kb audio for those of you with bandwidth limitations. Also transcriptions, getting more expensive by the minute, literally.

Steve: [Laughing]

Leo: You can get all of Steve's freebies, and you can get the latest SpinRite with a guaranteed upgrade to the next version. GRC.com. Next week it's a Q&A episode. That means you should go there and ask any questions you have.

Steve: Yes.

Leo: GRC.com/feedback is the place to post your questions, and Steve will pick 10 for next week.

Steve: And I think, since people are kind of - they're all asking me about perfect forward security and SSL. There's been a concern about whether the NSA is capturing traffic now for later decryption, and whether getting someone's private SSL keys at any time in the future would allow them to - would allow the NSA to decrypt all of the captured communications in the past. So in two weeks our topic will be Perfect Forward Secrecy and SSL.

Leo: Excellent, as always. You da man. If you want to watch this live, you can, 11:00 a.m. Pacific, 2:00 p.m. Eastern time, 19:00 UTC on Wednesdays at TWiT.tv. But we do make on-demand audio and video, high-quality audio and video available on our site, TWiT.tv/sn. And the best thing to do is subscribe. That way you'll always have Security Now! for your listening pleasure. Steve, we'll see you next time on Security Now!.

Steve: Thanks, Leo.