Windows Secure Boot

Description: Leo and I discuss the recent Pwn2Own hacking competition. We examine another serious breach of the Internet's certificate trust system and marvel at a very clever hack to crack the iPhone four-digit PIN lock. Then we take a close look at the evolution of booting from BIOS to UEFI and how Microsoft has leveraged this into their "Windows Secure Boot" system. We also examine what it might mean for the future of non-Windows operating systems.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-500.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-500-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here to celebrate his 500th episode with a look at the new Secure Boot. You can expect Windows 10 computers to be more locked down than ever. How does that work? He'll also talk about the latest security news, including a new entry on the Google Do Not Trust List. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 500, recorded Tuesday, March 24, 2015: Windows Secure Boot.

It's time for Security Now!, the show where we cover your security online. And we couldn't do it without this guy right here, the Explainer in Chief, Steve Gibson of GRC.com. Hi, Steve.

Steve Gibson: Leo, Episode 500.

Leo: Oh, my goodness.

Steve: Now, some people, notably Simon Zerafa, who's tweeted several times, saying, well, what about 512, you know? Shouldn't that be, like, a thing?

Leo: Oh, you binary bigots.

Steve: And it's like, well, okay, yeah, that's going to be good in 18 weeks. But right now,

Episode 500.

Leo: Well, you know what it does, it points up the fact that this is really just an odometer number. There's not really much, I mean, when you say we've been doing this show for 10 years, now, that's meaningful to me.

Steve: Right, right.

Leo: And that's coming up very soon.

Steve: Well, exactly, because you have some of your daily shows that are in the triple, or, what, in four digits already.

Leo: Giz Wiz is, yeah, Giz Wiz is in thousands. But our 10th anniversary, do you know, have we figured out when that is? Because that's in six months or so, I think.

Steve: I think it was shortly after TWiT. I think we started Security Now!, like in the late summer.

Leo: Okay. So soon.

Steve: So, yeah, maybe about six months, yeah.

Leo: Yeah, because our TWiT, April 19th will be our 10th Anniversary TWiT. That's going to be a fun event.

Steve: Nice.

Leo: Get all the original guys on the show. And I'm looking forward to that. So that's a special This Week in Tech on April 19.

Steve: So this week I was sort of moved to this topic because of some news that came out of Microsoft's WinHEC, the hardware engineering conference in Shenzhen, China about four days ago. And a slide that was shown, talking about the UEFI Secure Boot and TPM details, caught a lot of people's attention because, under UEFI Secure Boot, in order to get the Windows 10 Logo, which is the prized, what everyone wants to have on their laptops so that they can say we're an official Windows 10 Logo laptop or desktop or whatever, you must ship the machine with Secure Boot enabled. You must have UEFI v2.31 or later. And 2.31 is when Secure Boot essentially happened. It sort of appeared in 2.2, but 2.31 is now the standard.

Then they said, for Windows 10 Mobile, "Must not allow Secure Boot to be turned off on the retail device." Well, that's not a biggie. That's sort of like the iPhone. You don't want

the iPhone's boot integrity to be - you're not supposed to be able to turn that off, you know, it's an appliance. So the Win 10 Mobile is an appliance in that sense.

But what generated a ton of news stories when people saw this is that, for Win 10 desktop, for the first time, they said it's the OEM's option whether to allow the end user to turn off Secure Boot. Now, if this stands, and this is subject to change by the time it finally happens late this summer, presumably, but this is a policy change from Win8 because Win8 also used UEFI Secure Boot, had to ship with it enabled. However, in order to get the logo certification for Win8, the OEM had to allow the end user to turn it off and had to allow them to add their own security certificates to the UEFI Secure Boot database.

So this sort of, I mean, this has sort of always been Microsoft's approach is, for example, in XP they added a firewall. And this was still when the market was selling third-party firewalls; but, oh, not to worry, it's off by default. And sure enough, it really didn't affect anyone because it was off by default. But then, of course, famously with Service Pack 2 they turned it on. And so Microsoft sort of creeps along like this. Anyway, so the issue is that, if you were to purchase a machine which the OEM had removed the opportunity to disable Secure Boot in the BIOS, then that's a Windows appliance. You can't put Linux on it. You can't do anything else with it that you may want to.

Leo: You don't own it, really. It's not yours.

Steve: Well, it's sort of more like a big flat phone in that sense. So anyway, this stirred up a bunch of commotion. And I thought, this is a perfect opportunity for us on Security Now! to look at the technology of UEFI, what's that about; what's Secure Boot;, how does Windows interact with it. And down at the technical level with certificates and hashes and all that, how does that all work? So that's the topic for the show today, Windows Secure Boot.

Leo: Excellent.

Steve: And there was a little bit of news, really a perfect teachable moment for an iPhone/iPad four-digit PIN hack. I just loved the cleverness of this, so we'll talk about that. Another, an even worse certificate was found in the wild than we talked about last week. And I've heard you now both on MacBreak Weekly and on TWiT talk about the Pwn2Own contest, which I thought was just amazing because that kid who cracked all the browsers probably could name his salary.

Leo: Yeah.

Steve: Actually, he doesn't need a salary, given the prize money.

Leo: Yeah, $255,000 he won.

Steve: Yeah, yeah. So we have a great podcast today.

Leo: Excellent. I'm excited. I hope you are, too, since it is, after all, our 500th. And Lisa and I wish you a very happy 500th and our deepest prayer that you will do another 500 for us.

Steve: It's going great for me. I ought to also mention, just because we won't be talking again until after, I turn 60 in two days.

Leo: Happy birthday. That's a big one. I hope you're doing something fun.

Steve: Well, I've got all kinds of plans. So it's funny because, you know, like...

Leo: That's a huge birthday. Wow.

Steve: Yeah. And I guess what I like about it is that it was as I was approaching my five-oh 10 years ago that I'd just published SpinRite 6. It was cruising along. And I got sort of, as I was approaching 50, I got focused on health. And so that's when I really began my dive into learning about supplements and longevity and health and inflammation, and got into the whole issue of heart disease and cholesterol and, of course, famously Vitamin D. And then later the whole ketosis thing and all that.

And for me, as I turn 60, what's exciting for me is that I can literally, I can truly say that I feel 10 years younger today than I did 10 years ago. Like, I mean, I just - I've never felt better in my life. We saw that I got a cold a couple weeks ago, so I'm not impervious. But I just, you know, nothing hurts. I wake up feeling fantastic. I'm going to do some scans on the day after my birthday. I like to get my carotid artery ultrasound scan, and I did remove a little bit of plaque about five years ago that was initially there when I began this quest for health, and that's all gone now. I reversed that.

Leo: Wow, great.

Steve: Anyway, I'm just having a great time. So six-oh. And it does feel like a milestone.

Leo: Oh, yeah.

Steve: And I'm just - I'm glad that everything's going so well.

Leo: Or as Web4384 says, "It's only 32 in hex."

Steve: That's right.

Leo: Just remember that.

Steve: That's right.

Leo: Although I don't want to know what it is in binary.

Steve: No.

Leo: I'm sorry. He's got it wrong. It's 4C.

Steve: Four Charlie.

Leo: Four Charlie.

Steve: Okay. So...

Leo: Security news.

Steve: Security news. You want to click the second link in the first page of the show notes.

Leo: Okay.

Steve: FoneFunShop.co.uk, and there's some trailer stuff. This is a really interesting hack for the iPhone and iPads which worked through v8.1, but inevitably was going to be foreclosed on as soon as Apple realized what was going on. But the fact is, this was hardware, a hardware box, costs about 120 pounds, which until just November, which is when they went to 8.1.1, was able to crack four-digit PINs, if you had your iPhone protected by a four-digit PIN code.

Now, we've talked about how hard Apple worked to make that impossible, specifically the famous "Erase all data after 10 attempts" lockout, the idea being that you just can't guess four-digit PINs day and night because after 10 mistakes it does a - basically, remember, it throws the keys away. It doesn't actually have to wipe the data because the phone is encrypted, and all it needs to do is wipe the key which is required to be inline in the hardware that performs the on-the-fly encryption and decryption as data goes in and out of memory.

So get a load of this. What this hardware does, you have to open the phone up physically. In fact, there's a picture of it in use on the link above, that TechWorm.net link, that shows the screen popped open, I mean, the phone guts exposed, which has to happen. Then you stick some probes inside.

Leo: Oh, wow.

Steve: Now, what this does is - get this, Leo. It tries 0000, determines instantly whether it worked or didn't. And if it didn't, it does an instant power cut to the phone before...

Leo: To prevent the deletion.

Steve: Yes, to prevent the counting.

Leo: Ah.

Steve: So Apple made a little mistake, and this is why this is such a perfect example of, like, real-world security. Apple's programmers said, okay, so if they don't enter the PIN code correctly, and "Erase data after 10 attempts" is enabled, then we increment a counter which we store in nonvolatile memory so that we are continuing to increment it until they enter it correctly, in which case we zero the counter.

What these guys did was realize they could break power, disconnect the battery so fast after a guess failure that the phone was in the process of recording the guess, but wasn't able to complete that update. So just a brilliant hack. I just love this because this is the kind of way security breaches occur, where something looks absolutely solid until somebody really motivated looks at the same code everybody else has been looking at and says, "Wait a minute. What if we chop the power right here and then reboot the phone and then try the next PIN? And if it fails, chop the power, reboot the phone, go again." And it turns out that works.

So all Apple had to do is change the sequence. They pre-increment before they do the test so that they've recorded the pending failure in case it does fail, rather than post-incrementing. They did that. They changed that in 8.1.1. That's all they had to do to foreclose on this really cool hack. But until last November, even with the iPhone 6 and v8 and all of its various incantations getting up to 8.1.1, you could crack it this way, if it was protected by a four-digit PIN. If you were using the more complicated passwords, there was just, you know, there was no way it was going to guess. As it was, it could take like 111 hours, so many days to go through 0000 to 9999. But if you're law enforcement, and you needed to get into somebody's phone, and it was protected by a four-digit PIN, this would do it until a few months ago.

So I just love this as a perfect example of how clever the hackers can get and how something that looks innocuous can be even further hardened against this sort of attack. But you'd also have to argue that, boy, it's hard to find all of these. It just takes, you know, it takes bad guys poking at the things that are trying to be secure, acknowledging they're going to find some holes, and then plugging them and hoping you don't add new holes as you move forward, which of course is always the challenge.

In the news, we discussed last week the really interesting hack with Live.fi where Microsoft had not blocked all of the administrative email account names that certificate authorities might use to authenticate ownership of a domain, and as a consequence an enterprising person in Finland said, hey, what was it, hostmaster@live.fi. He set up that as an alias to his otherwise Live.fi email, and then he had a certificate issued for himself for Live.fi.

Well, it turns out there was another instance of that that was a little less reported, and that was somebody also did the same thing at Live.be, which was another one of these

Microsoft properties, but just in a different top-level domain. But the big news of the week, which caused Chrome to immediately reissue and update their CRLSet - remember I was talking about how - in fact it was last week. I'm looking up at my CRLSet monitor here, which was at nine. And you'll remember that I'm entry number five. You know, revoked.grc.com is - I'm at five. And since then there have been a few other problems. But Live.fi was nine because they added nine. And I imagine that Live.be was 10, and then something even worse was 11 because they're now up to 11 of these explicitly blocked certificates.

Leo: There seem so few.

Steve: Well, it is, because these are the ones which are not yet expired. So there have been horrible problems in the past, but those certificates have now expired themselves just by their date. So that's one of the nice things, I mean, I remember years ago grumbling about the need to constantly be issuing certificates, how they expired, and it was just a moneymaking enterprise for the certificate authorities. I have a much, I think, and I know that our audience does now, too, a more sophisticated and mature understanding of the fact that, yeah, that's maybe not ideal, but it's the best solution we have because, if certificates weren't expiring, that list which you note is so short could not be that short because every certificate that had ever had this problem would have to be on that list. Now it's only those that haven't otherwise expired themselves. So this is that tradeoff that we make where we really don't have a perfect system.

And the event that occurred which caused the 11th entry in that table is another biggie. It turned out, and we have to thank Google for this, I mean, one of the neatest things about Chrome is that Google has pinned all of the Google certs, which is to say the Chrome browser knows the serial numbers of every legitimate Google certificate, meaning that you can forge other people's certs; you cannot forge Google certs. And look at them with the Chrome browser, the Chrome browser will have a fit just instantly and immediately has instrumentation sending alarms back to Cupertino. And immediately...

Leo: Mountain View.

Steve: Mountain View. Immediately Google knows...

Leo: Those other guys are there in Cupertino.

Steve: Right. Immediately Google knows that one of their browsers somewhere in the world has just encountered a fraudulent Google certificate, and that happened again last week. It's happened several times. It's how we find out about these things so quickly is somebody messes with Chrome. And Chrome just - it's finicky this way.

So what it turns out happened was China's CNNIC, that's the China Internet Network Information Center, which is a large Chinese certificate authority, under an agreement with an Egyptian intermediate certificate authority called MCS Holdings, CNNIC issued MCS holdings an intermediate certificate that had signing authority, meaning that that certificate had the bits set in it that allowed it to sign other certs.

Leo: This was an error.

Steve: This, well, okay. So the agreement was that MCS Holdings would only issue certificates for their own domains. Instead, they broke their agreement with CNNIC, which is the root certificate authority that all of our browsers trust. I mean, this is like the equivalent of the Hong Kong Post Office. This is, you know, CNNIC is a CA, a root certificate authority all browsers trust. So their self-signed certificate is in all of our OSes and phones.

Leo: Is that China NIC? Is that what that is?

Steve: Yes.

Leo: All right.

Steve: Yes. So this CNNIC issued an intermediate certificate that itself could be used to sign other certs to this MCS Holdings, this Egyptian intermediate certificate authority, with the promise from MCS Holdings that they would only use it to sign their own domains. Instead, they installed it in a proxy.

Leo: Whoa.

Steve: So what they did was they stuck it in a piece of equipment that was then able to filter all of the traffic moving through it. Now, we've talked about this a lot because of course this is what some of the spyware that we've been talking about recently has been doing. But they didn't have a certificate that could sign other certs, that is, a certificate that was chained to a root cert that everybody already trusted. That's what's different about this. MCS Holdings had this certificate that could sign other certs, and it was trusted by the root cert that we all use.

Normally, when you want a proxy, like when the spyware is proxying, they have to install their public key in your browser in order for you to trust what they sign with their private key. But by installing - and this is the reason MCS Holdings presumably did this was they put this certificate in a proxy so that everybody downstream of the proxy would trust all of the security, and they'd be cracking open, basically performing a man-in-the-middle attack, cracking open secure transactions for whatever reason they had. But somebody inside of that network had Chrome.

Leo: Woohoo.

Steve: And the moment they went to a Google property, the Chrome browser received a fraudulent Google cert from this proxy appliance and raised holy hell and immediately exposed the fact that something was generating these fraudulent certificates. Google got on it, figured out what was going on, and has blacklisted that cert. Firefox will be updated. I got a couple updates from Firefox. There was a Firefox update, I think we're

at like 33 - let me take a look here, just see where I am.

Leo: They said it's going to ship in Firefox 37.

Steve: Ah, right. So I'm at 36.0.4. One of those changes was to fix the Pwn2Own vulnerability we're going to talk about next. But you're right, in 37 they will push out a Firefox that has this intermediate cert untrusted, much as Chrome now has, thanks to the CRLSet.

Leo: Now the interesting thing, the Mozilla security blog points out that the CNNIC, the China NIC, said as you said that it wasn't permitted by their policies and have revoked the intermediate certificate. But this is an issue because, as we know, certificate revocation is basically meaningless.

Steve: Yes.

Leo: So they have to add it to the list, that 11-member list, which includes Steve, of certificates you should not trust.

Steve: Right.

Leo: Firefox does the same thing. But if certificate revocation worked, this would have been automatic, and it's kind of annoying.

Steve: Exactly, yes. And this is where I kind of went off on my complete tirade many months ago, when I realized that Chrome was misrepresenting their own CRLSet because they're having to manually revoke highly important certificates by adding it to a special list at the top because their CRLSet isn't actually big enough to really do much good. And this highlights, again, essentially the problem that we talked about last week, which is it is - we have currently a brittle system. It's brittle because a tiny mistake that any one of the hundreds of certificate authorities that we implicitly trust, a tiny mistake they make requires a scramble. Suddenly all of the browsers and other trust anchors that exist on the Internet have to scramble in order to deal with the event of a tiny mistake being made. So this, I mean, it's just not stable the way it is. And the good news is there are things on the horizon that are going to probably fix this.

Leo: Interestingly, Mozilla will introduce with 37 a similar list. They call it, what is it, OneCRL, but it's the same idea.

Steve: Yup.

Leo: And they acknowledge that certificate revocation using OCSP doesn't work. So they're going to do a list just like Chrome does a list. So I guess, you know, maybe

that kind of confirms this as the best way to do it.

Steve: Yeah.

Leo: Till we fix it.

Steve: So at the CanSecWest security conference over the weekend - I have to say this guy is gifted. This was the Pwn2Own competition, or challenge, which is cosponsored by HP's Zero Day Initiative and Google's Code Zero program. I'm sorry, Project Zero program. Found a bunch of bugs. Microsoft Windows was found to have five, IE had four, Firefox had three, Adobe Reader three, Adobe Flash three, Apple Safari two, and Chrome one. Overall, a total of $442,500 was paid out to researchers. But more than half of that was grabbed by one guy. And he's a Korean researcher, is it pronounced Jung Hoon Lee?

Leo: Your guess is as good as mine on that one.

Steve: Yeah. He pulled off - okay. So as I just noted, IE, Firefox, Safari, and Chrome, all four browsers collapsed under the pressure from these guys. He was responsible for three of the four. And against Chrome he won $110,000, which is the single biggest payout in history. As you mentioned on MacBreak Weekly, he used more than 200 lines of code. This used Chrome's...

Leo: Oh, 2,000.

Steve: I'm sorry, 2000 lines of code.

Leo: Two thousand.

Steve: Two thousand lines of code.

Leo: Which is like writing an application. I mean, that's a big deal.

Steve: Well, and he made reference to the Chrome Native technology, which I'm going to have to cover on a podcast soon because it's really interesting. I took a look at it recently relative to exploit mitigation that they're doing. But this is the technology that is really sort of frightening, that allows you to run native x86 code at full machine speed, no interpretation, like any JavaScript or Java or Flash or anything else has, native code in a special sandbox in the browser. So it's very likely that 2,000 lines of this is what he used.

So he said, "Using more than 2,000 lines of code, Lee took down both the stable and the beta versions of Chrome by exploiting [get this] a buffer overflow race condition in the browser. He then used an info leak and the race condition in two Windows kernel drivers

to get system access." So the standalone Chrome bug fetched him $75,000. The privilege escalation bug got him another $25,000. And then Project Zero - so that would be $100,000. Then Project Zero independently paid him $10,000 when Chrome was hacked at that event. So just for Chrome, the Chrome takedown, $110,000.

Then in IE11 he earned $65,000 for exploiting a 64-bit version, this is IE11-64, with what's called a "time-of-check to time-of-use" - the acronym is TOCTOU, time-of-check to time-of-use - vulnerability. Who even knew there was such a thing? The vulnerability exploits the time between the time a file's property is checked and the time the file is used. This is another one of those little shim sort of things where, if you're really good, and you really understand this stuff, you can foil the system designed to be foolproof. And normally this would lead to a privilege escalation, but in this case the attack enabled him to gain read/write privileges on the browser, while another attack he used allowed him to escape the sandbox via a JavaScript injection which allowed him to evade other IE11 defensive mechanisms. So again, this is some serious aikido of this kind of hacking.

Leo: Yeah, almost all of the CanSecWest exploits were what they call "chained." That's a chained exploit, taking advantage of a flaw, flaw, flaw, flaw, flaw down the road. That's why 2,000 lines of code.

Steve: Yeah. And it's because one problem will get you a little bit of a beachhead, but nothing you can immediately exploit. So you have to take that and then leverage that into something else to get a more powerful advantage and then do it again. I mean, I feel like we're in the land of science fiction now. It's just amazing to me that this is all true.

But then, anyway, finally in Safari he found a "use after free" vulnerability which exploited an uninitialized stack pointer in Safari's sandbox in order to break out of the sandbox, and that earned him $50,000. So he took home $225 grand for a few minutes' worth of a demonstration. As we noted, and as you have said on both those prior podcasts, Leo, these are, I mean, it's a little bit of a mixed blessing because this 2,000 lines of code he didn't sit down and write right there. He had figured out that he was going to be able to crack Chrome. And in fact, in some background research I did, he did this through static code analysis. That is, he sat there looking through code in order to find a way to do this. So he earned his money.

And so I guess I'm glad that these vulnerabilities have been removed. It certainly does take this level of skill in order to find them. These vulnerabilities are increasingly difficult to find and exploit. And of course that's the good news for all of us. Browsers are getting more secure. Now the key, as I keep mentioning, is for us not to break them by adding stuff to them.

And so we have a Firefox update coming. I'm now at 36.0.4, and I was at .3 yesterday. So they've been revving it a bunch. I don't know whether this may already have the provision for that erroneous cert or not. I mean, it's not a huge problem. It's very likely the case that no malicious certificates were minted by this proxy. But it is very cool, I think, that somebody using Chrome downstream of the proxy visited Google, and that just sent Chrome into a fit and immediately identified that an illegitimate Google certificate was in use, and then we were able to figure that out. So this is some cool instrumentation that Google has been building.

Leo: No kidding.

Steve: And I loved Sunday's TWiT, Leo. I just wanted to...

Leo: Thank you.

Steve: ...mention that. And...

Leo: I don't remember it. What happened?

Steve: I completely agree with you about VR headsets and pornography.

Leo: Have you tried it?

Steve: I watched - no, no, no. And you are right. And I was a little disappointed that everybody else was just, like, in shock when you said that, as if...

Leo: They hadn't even thought of the idea.

Steve: Well, I mean, any student of history and technology knows that, for whatever reason, adult entertainment is always on the leading edge of new technology. The reason VHS videocassettes took off, initially, adult entertainment was what people were purchasing. And we are probably - everybody listening to this podcast is old enough to remember that in the early days, the Internet was mostly porn.

Leo: Right.

Steve: I mean, that's what - it was a huge...

Leo: May still be, but we just know not to go there.

Steve: Yeah, I haven't seen any in, like, forever. But, I mean, it used to be on sitcoms and things. I mean, it was a massive porn database. That's what it was. And people sort of had a sense of anonymity. And so that made them feel safe. And so, you know...

Leo: I actually think it's not only not changed, it's gotten - that really is probably a huge amount of the Internet traffic. And people just, I think, everybody just kind of pretends it's not the case or just ignores it.

Steve: Well, if nothing else, there is a lot more now.

Leo: Yes.

Steve: It used to be that, you know, there wasn't that much else on the Internet.

Leo: Right. There's a ton of other stuff, too, right. Everything's on the Internet, yeah.

Steve: Right, yeah, exactly.

Leo: Yeah. The porn guys were early on. They were VHS, every new technology. Credit cards, you know, online charges.

Steve: I'm sure the very first tintype photographs, it's like, oh, look at this picture, oh.

Leo: But my point - well, I don't know if I want to belabor it. But my point was, having tried it now...

Steve: The VR.

Leo: Somebody who has a Gear VR headset, which is based on Oculus, surfed to one of these sites that offer it. We didn't buy anything. They had trailers, so I looked at the trailers, which are explicit. And the experience is very realistic. It's notably more realistic than just watching it on a screen, on a flat screen. And while I haven't really enjoyed the realism of games - you get seasick because you're moving, and it's moving in a little different way, and it's just tough. And regular movies don't lend themselves, I think, so well to the idea of you're in the movie, certainly nothing that was made yet.

Steve: Yeah, you know, I mean, like what was James Cameron's big one, the...

Leo: "Avatar."

Steve: "Avatar." It was like, it was in 3D. It was nice.

Leo: Yeah, but VR the idea is that you're not just watching the action on the screen, you can look around; right? So that doesn't - moviemakers are not going to be that interested in letting the viewers' attention wander to any part of the screen.

Steve: No, no. They're artistes.

Leo: Yeah.

Steve: They want to control your focus.

Leo: And maybe they'll play with that. But nobody has to date. Well, as it turns out, I don't want to go into great detail here, but...

Steve: Well, but actually there's no need to.

Leo: You get it.

Steve: All I wanted to do was to, yeah, I just - listening to TWiT, I thought, what's wrong with you guys? Leo is right. It's obvious that that's going to be a market for virtual reality headsets. It's always this...

Leo: More obvious than you think, Steve, because it's a light years' different experience. It's like you're there. And that is what people want with porn. Not with many other experiences. Gaming, yes. If they can get it to work, great. But, boy, they've got it now.

Steve: It's a visceral, emotional, deeply wired into the human psyche thing.

Leo: Right.

Steve: And so, yeah, anyway, I immediately understood what you were saying, and I just wanted to say that those guys somehow, maybe they were just too shy. I mean, you know, we're adults.

Leo: I think they were absorbing it. But I'm surprised there's been nothing written about it. Nobody's talked about it. It happened with Google Gear very quickly, or Google Glass, rather, very quickly, that there were attempts to use it. That's not the same thing.

Steve: Yeah, a little window over here.

Leo: That's looking in a window over your eyebrow.

Steve: No.

Leo: Imagine an immersive adult experience that a lot of people would be very interested in. And it is not like - it is almost as...

Steve: It is really convincing. That's the point.

Leo: It's the thing this was made for, practically. And why nobody's written about this is beyond me because I feel like this will be a very big part of what makes VR happen.

Steve: I think it's just too soon. It's just, you know, they're still showing 3D cubes. It's like, oh, look at that 3D cube floating in space. It's like, well, no.

Leo: I remember going to SIGGRAPH 15 years ago, 1991 or '92, and they had - you could fly on a pterodactyl. Now, some of you may remember this. It was a very famous VR demonstration, required a lot of hardware.

Steve: Oh, my goodness, rooms.

Leo: And you're flying. And as you're flying, you can look around. That's cool. But if you were flying with a naked woman? I don't want to go any more. But I'm just saying...

Steve: No need. No need. I just, you know, it was just my reaction to them not understanding how obvious, I mean, just, if nothing else, the history. For whatever reason, that has always been a major application of leading-edge technology.

Leo: And you ain't seen nothin' yet. That's all I'm saying. It's just beginning.

Steve: And the other thing I wanted to mention about Sunday's show was it was interesting to listen to Ed Bott defend his position on privacy.

Leo: I give him that opportunity every time.

Steve: Yeah. And it's, I mean, and you and I sort of feel similarly. But that's the sort of - that's the feeling that I got when I came out of the Snowden movie, "Citizenfour," was a deeper sense of respect for the rights of people who wanted privacy for its own sake. I really think that that's where Ed is. He's like, he wants it for its own sake. He feels like he's always had control. He doesn't want to lose that control. And, you know, it's his right. And I think...

Leo: Of course it is. Nobody denies that.

Steve: Right, right.

Leo: And the only point I continue to make is that a lot of the services you use today, like Google and Facebook, are paid for by that trade of data for utility. And to me it violates - it's an ethical violation. And by the way, I think people think that I'm talking about TWiT. I'm not. We don't - we have kind of a different model anyway. But I guess it would be the same thing. If you'd never listen to the ads on our shows, but love the shows and listen to the shows every time, that's the same kind of ethical violation. You're saying, "I'll consume the content, but I will not support the way you pay for it." And that's, you know, it's fine if you don't want to participate. So don't. But don't use Google and Facebook and say, "But I don't want to see the ads." Because...

Steve: Although Ed did keep reminding you that it's the tracking that he objects to.

Leo: Yeah, yeah. And you don't...

Steve: Not the ads. Because, I mean, he really understands the incredible amount of data that is aggregated about us.

Leo: Yeah, I understand, too. Yeah.

Steve: And so that's the thing that he objects to.

Leo: And when you watch a show, we don't in any way track whether you saw the ad or not. We can't. We wish we could, but we can't. There's no way of us, when you get a show, knowing anything about you. We don't. So even when you watch live, I mean, there are a few trackers on our site for analytics because, you know, somebody pointed out, well, you've got a Facebook "Like" button on your site, which I think we do somewhere. That's a tracker. Of course, Facebook then is getting a bit of data about you, mostly just your IP address and maybe your browser, whatever they can deduce from the browser. Maybe they're setting a supercookie?

Steve: They know where you are, yeah, because...

Leo: Yeah, well, your IP address, yeah.

Steve: Well, no, they know you're at TWiT because your HTTP referrer header will say that there's a Like button on TWiT.tv. So they get that.

Leo: Yeah. It's basically giving Facebook a window into whoever visits our site.

[Crosstalk]

Leo: I'll take those buttons off. We won't have them on the new site. We will have analytics because we need to know how many people visit our site, not necessarily for advertising reasons, although that's part of it, but we want to know that. You know. And so if you're using an ad blocker, you're basically saying, yeah, I'm going to steal your stuff. To me. But I understand the other issues. The privacy issue is absolutely right on. Just don't visit sites. You can listen to our show. That's safe.

Steve: Yeah. So just a really short, nice little note from a listener of ours, Phil Horowitz, who's in Montreal, Quebec, Canada. He said: "Hello, Steve and Leo. My simple everyday SpinRite story: I was helping a friend who had a Windows 8 computer that hung at startup. He didn't know what to do. So he brought the computer over, and I ran SpinRite. As I have seen before, there were no evident errors that were corrected. But nevertheless, after SpinRite completed, the PC started. Great. Love Security Now!. Haven't missed a show. Phil Horowitz."

So, and I did want to take this opportunity just to say back to the listeners of this podcast on the occasion of our 500th podcast, I just wanted to say thank you because the support that I feel from our listeners, I mean, how many times have I read testimonials or tweets from people who have said, "I know the only thing you sell is SpinRite. I bought a copy, even though I didn't need it, to support you and the podcast. And I now know that, when a drive crashes, that I'll probably be able to recover it and save it." And that means a lot. I mean, it's one thing for people to say, "Hey, you know, love the podcast." But when people vote with their dollars, that's significant.

And for me - as Leo, you're always mentioning that SpinRite, as I just said, is the only thing that I sell - the podcast, this podcast has been one of the best things I've ever done for helping to get the word out about SpinRite. Prior to that, the reason I was doing the InfoWorld column back in those early days was that I was able to trade an ad insertion with InfoWorld for my column. They let me let the world know about SpinRite; I put a column worth of content in InfoWorld every week. And so that worked for me.

And for me, this is the same sort of thing. I love doing the podcast. But my ability to tell people about SpinRite for the last 10 years of the podcast has allowed the word to spread. It's helped innumerable people who wouldn't have known about SpinRite and would have lost data that they considered vital and important. And as I often say, it's paid all the bills. It allows me to have Sue and Greg to do the backend bookkeeping and tech support and gives me the freedom to push forward on other things. So anyway, I hear thanks from our listeners all the time for the podcast, which really means a lot to me. But I want to really turn that around and say thank you to our listeners for making this possible for me. Because your support and you guys all do that.

Leo: Thank you. We do. We agree. Awesome people. Keeping up the content flowing.

Steve: So, speaking of content.

Leo: Yes.

Steve: Windows Secure Boot. Okay. So this will sound familiar because it's very much

the same story that any system booting in a hostile environment would have. The reason it'll sound familiar is it's the Apple iPhone story. When we did the series, I think it was three podcasts, on I think it was iOS 7, all of the things that Apple did in order to really lock down, I mean, seriously secure iOS. They're booting that phone in a super hostile environment. Bad guys desperately want to get in. And the phone is sitting there all by itself, no one to defend it. It has to defend itself.

So there's now sort of a well understood approach to this. And it involves a so-called "trust anchor," that is, some absolute, single point of trust. And the evolution of what the device does, whether it's a phone, a desktop, a laptop, or whatever, all sort of flows from that first point of trust. So the original PC had the BIOS, an acronym everyone's familiar with: Basic I/O System, BIOS. And it was just about as simple as it could be. I meant to have a copy of it next to me.

I have the original IBM XT Technical Reference Manual, which is this really nice sort of clothbound, little sort of mini three-ring binder. And just innocently printed at the back of this technical reference is the listing of the BIOS. And it was really useful for me. It allowed me to create my first product for the PC, which was this crazy thing called FlickerFree, where I rewrote the BIOS handler for the display screen in order to remove the scrolling flicker that the CGA, the Color Graphics Adapter, card had, and also sped it up by hundreds of times. It was just amazing what a difference just rewriting a chunk of the BIOS could have. And similarly, in the early days of SpinRite, I needed to understand what was in the BIOS in the PC XT in order to interact with it because I was essentially taking the place of DOS.

Well, the idea with the original BIOS was very simple. The concept was the BIOS is going to be a layer between the hardware and the operating system. So that there would be hardware on the motherboard. You'd have serial ports, a parallel port, a graphics display, or at least a text display, keyboard, in the early days cassette input and output, and floppy drives, and maybe a hard drive. That was pretty much it. That was the I/O of the PC.

And rather than expecting any software that ran on that PC to talk to the physical hardware, what IBM provided was an interface layer, formally called a "hardware abstraction layer," an HAL, the idea being that, no matter what type of disk controller you might plug in, whether it was RLL, or SCSI later on, or MFM or whatever, you could always talk to different makes and models of hard disk controller through the BIOS, interrupt 13 of the BIOS, in order just to say read and write these sectors, move the head, format the track and so forth. There were commands you could give. Similarly, interrupt 10 was the video interface. And so you were able to say clear the screen, print this line of text, and so forth.

Well, the problem, as the industry evolved, people wanted to pierce that layer because they wanted additional function. I remember, for example, I replaced the video portion of the BIOS with a little TSR, Terminate and Stay Resident program, FlickerFree, in order to dramatically enhance the performance of the screen on that system. But famously, there were things that people started to want to do that you could not do through the BIOS. I want to say VisiCalc. Was VisiCalc on the PC?

Leo: Oh, yeah.

Steve: Or was it Lotus?

Leo: Well, VisiCalc was, and then Lotus. VisiCalc started on Apple, of course, yeah.

Steve: Right, right. And so in Lotus I remember they were programming the video RAM directly because they just couldn't afford the overhead of the BIOS. The BIOS could have allowed them to do it. But the things they wanted to do, scrolling quickly, horizontally and vertically; moving, like, highlight bars around the screen. It just couldn't do that. So they had to bypass the BIOS and go direct.

So many older systems are still BIOS based. The BIOSes have lived for decades, mostly because you could bypass it. You would use the BIOS to essentially power the system up. It would initialize the hardware. It would sort of settle things down. Then it would look through a list of possible boot devices, checking them in sequence for a sector that said it had access to a bootable partition, and it would go and see if it could boot the partition. If so, it would run that code, and off you'd go. The operating system then, rather than using the BIOS - for example, DOS actually did use the BIOS. But the first thing Windows did was say, okay, fine, get out of the way. And Windows brought its own drivers, essentially to talk directly to the hardware.

So this was the situation up until probably, what, the mid-1990s or so, when the BIOS began to show its age. Systems were evolving. We were beginning to want much more capability. People wanted to be able to boot their system over the network remotely. They wanted, corporate IT wanted to be able to do an inventory of what was plugged into the motherboard without even talking to the operating system, actually have the motherboard be smart enough. Motherboards started to want to be able to monitor the voltages of their power supplies, and the current. They had multiple fans, and so they wanted to control temperature in various areas. They had fancy RAID arrays that they needed to support.

Essentially, the very modest platform that the PC XT originally was, that the BIOS was able to service, that platform just exploded. So we needed something new. And so the so-called EFI then became the unified extensible firmware interface, UEFI, which is now the state of the art in firmware. Some people say the "UEFI BIOS," although technically that's not right because the BIOS is the BIOS, and UEFI is a different firmware than the BIOS firmware. But today's UEFI offers a vast array of services. There's essentially almost an operating system within the motherboard to manage the modern complexity of all the peripherals. There's an ACPI which is the power control that allows various power-down states. And all that has to be communicated and coordinated with the hardware so that the motherboard understands how to do that to all of its hardware.

So you still need sort of a central point of responsibility. And as I mentioned, there's fans and voltage and current monitoring, remote network booting, just all the chassis management and everything. So the UEFI has just exploded in size. It's really left the original BIOS way behind. So it got to the point where it was time to talk about security. And the UEFI has gone through a number of versions. It was at 2.2 that we first really got what was known as "Secure Boot." And in the same way that the iPhone depends upon its hardware in order to provide absolute security, the Secure Boot is the same way.

There is a platform key, abbreviated PK, which the manufacturer of the device - we'll use the term "motherboard" just for short, but it could be also the main board of the laptop or whatever. There is a platform key which the manufacturer is able to use to sign the firmware which first wakes up when power is applied to the device. So essentially everything we have learned about the way certificates and public keys and all of the PKI,

the public key infrastructure, works, that is all there in the UEFI system firmware. The manufacturer has the private key that it probably doesn't let go of. There are some instances where really large purchasers may acquire the private key for their systems if they want absolute control over the firmware of the systems that they're purchasing and managing, although it's probably not necessary because of the hierarchical nature, sort of in the same way that you can get a certificate to secure your server. You don't need to be a certificate authority yourself.

So essentially, if we sort of reuse the jargon of the web, the manufacturer of the motherboard, this UEFI motherboard, is the certificate authority. It is the trusted CA for their device because, burned into the ROM of that board, unmodifiable, is a public key which is used to verify the signature of the first startup boot firmware that wakes up when the system receives power. So at a hardware level, before anything happens, the signature, the digital signature, in the sense of an SHA-256 MAC signature, is taken of the very firmware to start, and verified against this platform key. And of course so what that means is that only signed firmware will be booted by that board. If anything changes, if a byte changes in that firmware, or someone, anyone tries to replace the firmware - and in fact there are also rollback provisions, using timestamps, so that it is not possible to put an earlier version of firmware on top of a later version.

So the system has been designed in very much the same way the iPhone was, with a security framework that starts with absolute hardware support and then works to never lose that. There are three databases that are contained in nonvolatile RAM or ROM, doesn't really matter. It won't lose its memory when it's powered off, but it can be rewritten. Thus these are databases. There's something called the KEK, the Key Exchange Key, database, which contains essentially signatures, the spec refers to them as "trust anchors," but they're just cryptographic signatures of entities that are allowed to modify the other two databases. There's an "Allowed Signatures" database and a "Forbidden" database. And either of those databases can contain either certificates that sign other firmware, or hashes, that is, digests, fingerprints of other firmware.

So what we have is an architecture where all of the firmware modules, that is, for example, in UEFI you still have things like Option ROMs to extend the knowledge that the base firmware has of specific peripherals. So, for example, a network hardware will have its own Option ROM with UEFI-compatible firmware in it. In order for that Option ROM to be initialized at boot, which has to happen in order for it to initialize the network hardware, that Option ROM has to either be, well, first of all it has to be signed. And the signer of that has to have their certificate in the Allowed Signatures database and not have that certificate in the Forbidden database. Or if there isn't a certificate for the firmware, then the cryptographic hash of that firmware has to be in the Allowed Signatures database.

So you have an explicit whitelist/blacklist system for every component of UEFI firmware. All the various pieces of additional firmware extension that exist on the motherboard are individually hashed, and in some cases signed, in order to allow that firmware to - and I should mention that the firmware is enumerated when the system powers up. And then that initially signature protected initial boot code goes out and checks the signatures of all of the other pieces of firmware. As it's doing this, it's also leaving an audit trail.

There's another component of this which is oddly named. It's called "Measured Boot." So we have Secure Boot, which is sort of the implementation side of making sure that nothing that is known bad or unknown is ever allowed to run. And then the Measured Boot is an auditing system, an auditing function which is going on throughout this entire process in the background, which uses the Trusted Platform Module as its audit store because in a compromised system the compromising software could alter the audit trail.

We often hear, for example, how bad guys get into the system and then erase the logs of them getting in, in order to cover their tracks. So you want to prevent something like that from happening.

So this Measured Boot is also running step by step through the entire boot process, creating essentially an audit trail of everything that is run. So at some point, after all of the UEFI firmware has been enumerated, all of the separate pieces of it have had their signatures checked, certificates have either been found for them, or they are explicitly whitelisted in the allowed signatures database, finally then the system will enumerate boot candidates, like mass storage boot candidates, and begin to turn this boot system over to the operating system that wants to run on top of all of this.

So that's where, of course, Windows comes along. Windows is able to have, well, first of all, as we know, 64-bit kernel drivers have to be digitally signed. So the UEFI firmware is able to reach up into the Windows boot loader, which is an EFI image also. EFI images are Microsoft format portable executable files. And that's just a standard for UEFI. So UEFI is able to reach up, verify, in the same way that it verifies its own firmware, it's able to verify the first boot modules of the operating system also, to make sure that they are explicitly whitelisted and that they have not been changed.

Then Windows introduces - the operating system itself has this notion of boot drivers, which are kernel drivers flagged for early loading. It's just a particular property that the driver has in its header which says "I need to be loaded early in the process." There's a special one of those that supports Windows Secure Boot which is called ELAM, E-L-A-M, which is the Early Launch Anti Malware. And Microsoft uses the acronym AM for antimalware throughout their spec of this. So the idea is that the EFI, the UEFI firmware verifies the signature on these drivers. The drivers are signed by Microsoft Authenticode. So, and that technology is also available to the UEFI firmware. It's able to verify Microsoft Authenticode signatures in order to know that it's able to trust these pieces.

This ELAM, the Early Launch Anti Malware, gets in, inserts itself into the boot process, and then essentially takes over responsibility. That's the handoff between the UEFI and the Microsoft boot process, where Microsoft is now able to know that, up to the point that it has received control, only signed and trusted modules, from the first moment power started to flow through the motherboard to now, have been able to operate. Microsoft's own boot system then pulls in the balance of the pieces of Windows, verifying every piece of them, and also doing antimalware checking against various types of dictionaries that it has available as this progresses.

And the final thing it does is it looks in the Trusted Platform Module for this Measured Boot audit, and it has the ability to send that out of the machine to a remote server where that audit is verified. It understands that, if something has gone wrong, it can't be trusted to audit itself. But the TPM is able to produce its own signed audit trail, which cannot be altered. So that can be sent out to a third party in order to verify the security of the boot.

And so, for example, there's another term, "Trusted Boot," which technically is the combination of a secure boot and an audited boot, which is this Measured Boot technology. Together, they allow Windows to get itself up and going in a trusted state. And, for example, it might be, in a large enterprise, that that computer is not allowed onto the network until that measured boot that is the audit has been verified by some other machine on the network, confirmed that this system is up in full trusted mode. Every piece of it that has run so far is trusted. There's no malware present. And then the machine is given permission to get onto the corporate network.

So essentially what we've got is a rather straightforward, although it is really complex, the specification. I was looking at - it's now at 2.4 Errata Level E, I think. It is a 3,000-page - it was a 2,998-page specification. And, I mean, it's just unbelievably complicated. But it's because you've got this, you know, everybody got their features in. And the fact is, I mean, UEFI is like a world unto itself. It is a whole operating system. There's a command language. You can issue commands. You can get consoles. I mean, it's amazingly sophisticated. Basically, all we normally see is, you know, we turn the computer on, and up boots our operating system. All of this stuff goes on hidden in the background.

So understanding what this is, we can imagine what it means if we could not turn it off. Because if you own a computer, and there is no way to turn this off, to turn off Secure Boot, then you have an appliance. You can't change the operating system. You can't mess around with the computer at all. It is, I mean, you can do things that it allows you to do, but nothing that it doesn't because, from the first moment this thing receives power, everything has to be signed and trusted.

Now, this is why, because this was sort of a controversial move, why the Windows 8 Logo requirements, as I mentioned before, explicitly said that Secure Boot must be enabled when a new machine is shipped. But it must be possible for the user to manually disable. I should also mention that there is a post-boot programmatic interface between the operating system and the UEFI. That is to say, UEFI exposes a large array of services for managing all of these trust databases which Windows, for example, knows how to talk to. And that's necessary.

For example, when we're updating kernel drivers, they're probably going to all be signed by Microsoft certificate, so they would be trusted anyway. But you might have, for example, a third-party driver which Microsoft needs to bless and, for example, put its signature into the trusted UEFI database when Microsoft installs that driver into Windows. Or at least Microsoft wants to verify that this driver that it has installed into itself to be booted will be trusted by UEFI at boot time so that UEFI will allow that driver to load. So there has to be a post-boot interaction between the operating system and basically the crossing guard, you know, the guard of the bridge which is sitting there deciding who gets to run or not.

So the controversy is that, whereas with Windows 8 Microsoft said systems must be allowed to allow the user to turn that off - and that's the key. You cannot programmatically turn it off. There has to be no way for malware to turn it off because that's what "programmatic" means is that malware could flip that switch, then cause the system to get rebooted, and then gain an early foothold. Basically what we're doing is we're preventing any kind of firmware hack or bootkit or rootkit-style exploit, which has always been the Achilles heel. We've talked about how the operating system can't be responsible for what happens before it gets in control. It can do the best job it can to prevent anything bad from happening once it's in control. But before it's in control it's not there. So if something is able to modify it before it runs, then it's already corrupted by the time it becomes aware and is able to operate.

So this whole system is about preventing messing with the firmware and messing with the operating system as it loads up to the point where it's then able to do the best job it can of assuming responsibility for anything that happens subsequent to that. And the unknown is how the systems are going to look, the Windows 10 Logo certified machines, where Microsoft has specifically said OEMs can choose whether to allow users to disable Secure Boot or not. If that option does not exist in the BIOS, then essentially you've got Microsoft Windows installed...

Leo: And that's it.

Steve: ...and that's it, baby.

Leo: But I think there's a...

Steve: You can upgrade.

Leo: I think it's nice that you have the choice, so you don't have to buy that. You should find out if the OEM is doing that. It is completely appropriate that people should be able to buy a machine that is locked down. You know, I think about the Chromebook Pixel, which is Google's thing.

Steve: Yes.

Leo: Now, they're not using UEFI. They're using Coreboot, which is an open source...

Steve: Yup, yup.

Leo: And actually someday I'd love to hear what you think of that compared to UEFI. But in the Chromebook they have a very, I think a nice solution. You can disable it. You have to reboot into developer mode. It puts a big thing up that says, you know, you're now going to be able to run unsigned software.

Steve: Good.

Leo: But if you want to do that, you know, but if you want to back out, press the spacebar. So for most unsophisticated users it will always be - and they have a TPM module in there. It is secure. It's signed. They've made it a very, I think, I'm not an expert, and I'd love to get your opinion on that, but a very secure platform.

Steve: No, what you described in terms of the user experience I think is exactly right. I would opt for, I mean, absolutely have this thing be secure. But something feels, I don't know, a little creepy.

Leo: You bought the hardware.

Steve: Yes.

Leo: You should be able to modify it.

Steve: That's what it is, exactly. It's like, you know, I can't do anything else with this? It's like, I mean, it does turn it into a big phone, essentially, that I can't do anything with. That's like, you know...

Leo: We see the need for it. We talk about it every day on the shows.

Steve: Yes, yes. And for a lot of people, for a lot of people, I love the idea that we've got this level of security. Most people can't even get into the BIOS. They don't know where it is. I mean, I have a hard time. Is it Delete, or is it F2, or like...

Leo: And they change it, don't they.

Steve: It's difficult to get in. So it's already hard enough. And if they simply say, "You almost certainly never want to turn this off," I mean, put up a flashing red screen and say, "If you turn this off, we're no longer responsible. We're going to find out you did, and you no longer get support." It's like, "You're on your own." I think that's the way to do it. But it'll be interesting to see. Now, I'm wondering politically what's going on. You think this is OEMs asking Microsoft, please don't require us to allow the end user to disable Secure Boot because some of our customers have on Windows 8, and they've gotten themselves in trouble. We really don't want to have to do that. Or is it Microsoft saying, eh, Linux. Don't think you need Linux.

Leo: I'm pretty sure it's Microsoft. I think you nailed it at the beginning where you say Microsoft eases into these things.

Steve: Creeps, yes.

Leo: And they kind of telegraphed this. And you may remember when Windows 8 came out there were a lot of Linux enthusiasts saying, oh, my god, this is terrible. They found out you could, yeah, you could disable UEFI. You can put Linux on these machines. But that, I believe, was Microsoft announcing to the world, hey, we're going in this direction. And I don't think it's - I don't disagree with it because - as long as some OEMs offer machines that you could put Linux on. But the problem is somebody...

Steve: Any system...

Leo: Or anything else.

Steve: Any system where you're going to install the operating system.

Leo: Right.

Steve: It's going to be UEFI today, and it'll have that option.

Leo: Right. There has to be.

Steve: Right. And if you install Windows 10, I'm sure Windows will say, hey, turn on Secure Boot. Everything's set. The databases are established. All the trust anchors are in place. Turn on Secure Boot, and you get to have your system locked down. You don't have to worry about rootkits and bootkits and firmware crud getting in.

Leo: Well, exactly, exactly. And as we live in this world of increasing threat, I think the mass of the market is very much interested in that as a solution and is never going to install some other operating system. They never - they installed Windows in the first place.

Steve: Nope, nope. Just give us a choice for - in those particular situations.

Leo: But for us - yeah.

Steve: Yes.

Leo: And that's, I have to say, one of the things I really like about the Chromebook Pixel, you can install Linux on it.

Steve: Nice.

Leo: And but you have to go put it in this insecure boot mode. And they're very clear about what you're doing. And if you're smart enough to do it, and you know the commands, then do it.

Steve: Now, if you...

Leo: It's funny, I run it - for a while I put crouton on it, and Linux. And, you know, I don't need that. It's just messing it up. I want to know that this is...

Steve: Crouton?

Leo: It's called "crouton" because it's a chroot script. It turns on a chroot virtual

machine. And then you put Linux in it. And it's, like, very elegant, actually. It's a clever hack. "Elegant" may not be the right word. It's a clever hack. But I like the idea that when I'm using the Pixel I know exactly what it does. It's absolutely secure. I just don't have to think about it.

Steve: It's worth noting also that, even though this would give Richard Stallman a seizure...

Leo: Oh, he'd hate it in every form.

Steve: Oh, my lord. It is possible to do certificated drivers with Linux. So it is often possible to still be in a non-Windows OS and use Secure Boot. You just have to go through the extra effort of getting stuff signed and certificates installed. But entirely possible.

Leo: And, by the way, I'm told in the chatroom, and I believe this is true, that Stallman does endorse Chromebook because that is an open source, free SFS program. What about SpinRite? The way I use SpinRite right now, I boot to a new operating system.

Steve: Correct.

Leo: FreeDOS.

Steve: Correct.

Leo: What are you going to do about that?

Steve: So if it turns out that we end up in a world where we're in Secure Boot, then I'll just do a version of SpinRite as a kernel driver because that's something I've always been thinking about. And that would allow it to run. Because I have an Authenticode, you know, all of my software is signed by Microsoft Authenticode. It's all Gibson Research Corporation, and we're trusted. And so I'm able to do kernel drivers. And a kernel driver has all the capability that I would need. So it may be that that'll end up being another piece of SpinRite that comes along, if it turns out that there are people who need SpinRite to run in a Secure Boot environment.

Leo: It does mean that - somebody's saying in the chatroom, and I think this is true, used hardware has reduced value. You know, remember one of the solutions to running an old XP machine was put Linux on it. Well, that won't be an option on old Windows 10 machines in many cases. It makes them more disposable, really.

Steve: And relative to SpinRite, remember that the story I just read, Phil Horowitz's

friend had a Windows 8 computer that was hanging at startup. If that was a Windows 8 computer, it probably had Secure Boot. He turned that off, and then he booted SpinRite, and it worked just fine.

Leo: I presume the system recovery boots would still work.

Steve: Oh, yeah. In fact, there is some fancy stuff Microsoft does. I mean, it has all this sort of multilevel fallback stuff where, if it gets to the point that it thinks it's okay, and it looks back at that Measured Boot audit and sees something fishy, it's able to immediately abort its own boot and fall back to recovery mode in order to, like, restore drivers and things. So there's all of this magic scary stuff going on that we hope works right. Although it seems like sometimes that stuff has a problem, too.

Leo: It's only a matter of time before somebody figures out something.

Steve: Yeah. So essentially, we've seen the evolution of our hardware platforms, along the same line that we followed Apple with the iPhone, where in order to put up the level of resistance to hacking that we really have to have in this day and age, the system has no choice but to essentially whitelist every component that it's going to load. And if you're going to whitelist every component, then you're going to be rigid against non-whitelisted components or operating systems.

Leo: I love this show. You learn so much from this show. If you enjoy it, I do hope you will listen each and every week. Subscribe, that's the best thing to do, and you can do that at TWiT.tv/sn or using your Podcatcher or iTunes or whatever you use to listen. Almost all of them will have a subscribe option. You can also listen through our great third-party apps. We have wonderful developers on iOS, Android, Windows Phone, Roku. I bet you there's BlackBerry. There's certainly Windows and Mac that you can always do it that way, too. But please, make sure you tune in every Wednesday at 1:30 Pacific Daylight Time. I say that because I think there's still some error in our calendar on the website, and I want you to understand we've moved to Daylight Time.

Steve: We've also moved to Tuesday, so...

Leo: Did I say Wednesday?

Steve: Yeah.

Leo: One of these days. It's only been six months. Tuesday, thank you. And so that is 1:30 Pacific Daylight Time, 4:30 Eastern Daylight Time. If you're on UTC, that's 2030, or you can make the calculations yourself. Most other countries are going to summertime soon, I think. So this won't be as much of an issue. There's just this little interregnum that is a problem. You can get - Steve's got 16Kb versions of the audio at his site, as well as full text transcripts that are great. That's GRC.com. You'll

also find SpinRite there, world's finest hard drive and maintenance utility, plus lots of other freebies. Steve's giving stuff away all the time. Did you hear our conversation, was it on TWiT, about Microsoft's new Passport and - hello.

Steve: And I appreciated your little mention of SQRL. You said, well, you know, Steve's got this SQRL thing he's working on, and we'll see. Yeah, we'll see.

Leo: Microsoft's supporting FIDO.

Steve: Yeah. We'll see how that goes. There's a lot of problems with FIDO. For example, FIDO doesn't have any identity management at all. That is to say, if your identity escapes from you with FIDO, there's nothing, there's no mechanism in FIDO for remediating it, for recovering it, for pulling it back. All of that is built into SQRL. So this is why Brad Hill, who was one of the chief architects of FIDO, said he thought SQRL was the best-thought-out system that he had seen. So, but again, you're right. It doesn't have everybody behind it. But it does have a lot of small guys behind it, a lot of - I'm getting email from people saying we want to support it on our website. We want to use it here or there. All the various platforms are going to be supported. We'll have SQRL clients available. And we'll just see. It might very well live side by side. I would have no problem with that at all.

Leo: Yeah.

Steve: And for one thing, FIDO is all based on hardware. And, you know, for example, you guys were talking about, and I really liked the idea, that you need this 3D vision in order that a photograph won't fool your face recognition in Windows. It actually needs to be a 3D representation, taken from multiple angles at the same time, to know that you're not putting a flat picture up in front of it in order to cause someone to be able to log in. So all of the biometric stuff is part of the FIDO spec.

And one of the advantages is that SQRL doesn't require any hardware. You're able to have a software-only client, which is absolutely secure. So it's also free, and you can add it to existing systems. For example, none of the hardware right now that Windows 10 will run on is currently available. You don't have 3D cameras unless you add that. So anyway, we'll see. I'm going to get it done. I'll get back to working on SpinRite 6.1. And we'll give SQRL a good launch and see how it takes off.

Leo: Find out more about SQRL at his website. Questions next week, you think?

Steve: Yup. Let's do a Q&A.

Leo: GRC.com/feedback. I suspect Steve's going to say this, but you correct me if I'm wrong. Don't put birthday greetings there.

Steve: Yeah, there's no need. We all know that I'm getting old.

Leo: I think that you probably would not like hundreds of those. Is there somewhere? Should people tweet you if they want to say happy birthday?

Steve: Absolutely. That would be very nice.

Leo: It's the big six-oh for Steve Gibson, @SGgrc. GRC.com/feedback if you've got questions for next week.

Steve: Yeah, you know, I could see us going to a thousand episodes because, you know, this last 10 years have been - it's been good, Leo.

Leo: I don't know about 10. Yeah, I can give you about 10 more, too. We'll both be almost 70. It'll be...

Steve: I do think that Pournelle is probably on his last legs. He was...

Leo: He's in his 80s, late 80s, yeah.

Steve: Is he really?

Leo: Well, I don't know exactly how old he is, but...

Steve: He's got so much wisdom. I love Jerry's wisdom.

Leo: Love Jerry.

Steve: But, boy, he's been battling.

Leo: Poor guy.

Steve: Yeah.

Leo: But you know what, he's doing fine. He got back from the stroke. He was on TWiT a couple weeks ago and was fabulous.

Steve: Yeah, yeah.

Leo: The mind continues. It's the body that falters.

Steve: Right.

Leo: If you're lucky. Or [indiscernible], I don't know. GRC.com. Thank you, Steve. We'll see you next week, right here on Security Now!.

Steve: Thanks, Leo.