Schneier on Equifax

Description: This week we discuss why Steve won't be relying upon Face ID for security, a clever new hack of longstanding NTFS and Windows behavior, the Vault 8 WikiLeaks news, the predictable resurgence of the consumer device encryption battle, a new and clever data exfiltration technique, new antimalware features coming to Chrome, an unbelievable discovery about access to the IME in Skylake and subsequent Intel chipsets, a look at who's doing the unauthorized crypto mining, WebAssembly is ready for primetime, a bit of miscellany, some closing-the-loop feedback with our listeners - and then we share Bruce Schneier's congressional testimony about the Equifax breach.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-637.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-637-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Lots to talk about, including a handy-dandy way to hack any Intel computer using the USB port and a look at Bruce Schneier's testimony in front of Congress about the Equifax breach. It's all coming up next, plus a great Image of the Week, on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 637, recorded Tuesday, November 14th, 2017: Schneier on Equifax.

It's time for Security Now!, the show where we secure you. We fasten your seatbelts, we put on your belt and your suspenders, and the tinfoil hat is optional. This is the guy, Steve Gibson of GRC.com, who helps us each and every week. Everybody loves Steve, loves this show. It's true, it's true. Fastest growing show on the network, partly because of security. But I think, you know, there was an article, Steve, I read about a basketball podcast called "Dunc'd On." And it's just two guys talking about basketball. And Bloomberg was baffled by its success and the fact that it has 50,000 downloads a week, and it's two hours long, and they talk obsessively about the details of stuff that nobody cares about in basketball. And I thought, well, god, if you'd listened to this show for the last 13 years, you'd know. That's what people want in a podcast.

Steve Gibson: Yes, yes.

Leo: Hello, Steve.

Steve: And interestingly, the topic of security never leaves us wanting.

Leo: No.

Steve: And we've got a fabulous Picture of the Week. The main topic for today's podcast, and the title, is Schneier on Equifax. Bruce testified in front of Congress last week.

Leo: Oh, good.

Steve: And what he had to say was really interesting. He presented 11 points, and I'll go over about the first half of them at the end of the podcast, then kind of summarize the rest. But he made some really, I think, valid points. And I'm so glad that he had the opportunity to explain what he feels is going on. And there's a lot of stuff we already know. But Bruce is in the middle of this and spends a lot of time thinking about this. And so he made, I think, some very salient and clear points. So we're going to get to that at the end.

But first we're going to talk about why I won't be relying upon Face ID in the new iPhone X for its security. A clever new hack of longstanding NTFS and Windows behavior, which many AV companies are now scrambling to fix, and I think six or eight have. WikiLeaks has moved from Vault 7 to Vault 8 with an interesting disclosure of CIA source code which reveals the structure of, if we are to believe this, the CIA's command-and-control system for their publicly implanted, I think "spyware" is the right thing to call it. Everyone calls it "malware." But if it's the CIA, is it malware? It's spyware, probably.

Anyway, so that's really - and we actually see the architecture of that, the "Hive," as it's called. We also have the predictable resurgence of the consumer device encryption battle after Sunday before last's Texas church shootings, and some interesting reaction from Apple and the FBI, and some commentary that's worth sharing. A new and clever - you're not going to believe this one - data exfiltration technique.

New antimalware, three new antimalware features coming to Chrome in the next couple months. An unbelievable discovery about a way to access the much-maligned, and deservedly so in some cases, Intel Management Engine in Skylake and all subsequent Intel chipsets. A look at who's doing the unauthorized crypto mining across the web. A security researcher looked carefully at the crypto mining that was going on, which by the way has exploded. It's four shy of 2,500 sites are currently doing it. But it's not who you would think.

Also, a technology we discussed 2.5 years ago, WebAssembly, which is a next-generation replacement for JavaScript, is ready for primetime as a consequence of the final addition of the last two browsers, which was Safari, which iOS 11 brought to us, and the Fall Creators update of Edge in Windows 10 now offers it. And we'll talk about that. And what's interesting is that you couldn't do cryptocurrency without it because browsers wouldn't be fast enough. But thanks to WebAssembly, they are. But we all have gained benefits of other sorts. We'll talk about that.

A little bit of miscellany, some closing-the-loop feedback with our listeners, and I want to talk about Schneier's congressional testimony about the Equifax breach. Oh, and a really fun Picture of the Week that a number of our listeners sent me, thinking that we would get a kick out of it, and indeed. Our Picture of the Week will mean a lot more to old-

timers because I don't know how many hours I have spent staring at the screen of my computer while it defragged my hard drive. There's something about it. It's just mesmerizing. It's like watching your car go through the car wash tunnel, where you just, if the car wash tunnel has windows, there's something fascinating about watching it just sort of go through without you and coming out clean the other end.

Well, defragging, of course, is the process of pulling all of the various bits of a file together in one place, which back in the day when hard drives were a bottleneck and when our file systems sort of cared more than they do today, or before we had SSDs where physically putting the pieces of a file contiguously on the physical hard drive demonstrably, maybe, increased its performance. If nothing else, it did decrease the wear and tear on the drive because the head didn't have to jump all over the place grabbing all the little pieces of a file, and files tended to fragment.

Anyway, the point is, those among us who remember the days of defragmenting our drives, you could still get defragmenters, but they're way less used these days than they used to be. Anyway, the picture is just very clever. The caption is "I defragged my zebra." Yes. And it shows the result of what you would get if you defragged a zebra. Normally they're striped and zigzag-y and all kind of complex, I guess for a form of camouflage out in the wild. Not after the zebra has been defragged, in this case. The front is black and the back is white because all the bits have been aligned with each other. So anyway, a bunch of our listeners found that and thought we'd get a kick out of it. And indeed, I wanted to share it for that reason. Very clever.

So, Leo, predictably, as the world has had access to their iPhone X with its Face ID facial recognition, we're beginning to have the claims that Apple initially disclosed tested, which is that it is, what, whereas there was a one in 50,000 chance, they said, of somebody, of a stranger unlocking your Touch ID fingerprint recognition, they were claiming a million to one. Now, maybe if you took, I mean, first of all, I don't know how you make that claim without testing it, and it would have been difficult for them to have tested it. So they may believe that's the case.

But what we're immediately seeing are coincidental, close-enough faces, for example, twins or lookalike family members, in some cases the son of a mother who looks enough like his mother to unlock her phone. And then there are the deliberate security researchers messing around with the technology, essentially since Apple hasn't told us in detail. They've given us lots of gee whiz-y stuff about each phone having its own unique sequence in which it throws out the IR array to scan the images and so no fixed system can be the same. It's like, wow, wow, that sounds like a lot of high tech there. But with some work, this technology has been spoofed.

And, I mean, I liked a couple of things that you were saying, well, that the group on MacBreak Weekly was saying beforehand. And a point that I had intended to make, Andy touched on, which is there is a difference between faces and fingerprints, which is that faces are generally far more public than fingerprints. And, for example, we know that there's a technology available to take multiple angles of a person's face and reconstruct a 3D model. So we also know that attacks never get weaker, they only get better. And we credit Bruce Schneier, who's the topic of this week's podcast, his congressional testimony, to saying that. And it's something we often repeat here because we see it happening in the world.

So what I believe is that Apple has made a huge concession, and an unfortunate concession, a concession giving up security, true security, in favor of convenience. So I'm not judging that. I mean, I recognize, I've listened to you talk, ever since you started to use your iPhone X, about just how incredibly convenient it is. And I get that. I could not

possibly use it because, for me, I have LastPass, which authenticates with an authentication app, and my authenticator app there. And essentially, somebody who got into my phone, because it's the portal to things that are really security sensitive, they could just take down my life. I'm statically logged in in Safari sessions. The authenticator is there, access to which would defeat its purpose, and so forth.

So I was encouraged - I don't have mine yet. I was encouraged when Rene said there is granularity that can be applied to what things it can be used for. And this immediately made me think, okay, well, I need an authenticator app which is independently lockable, and LastPass needs to be independently lockable because of course it contains the keys to the kingdom. I don't want to not have those things on my phone because that's the whole point. So what this makes me feel like is that maybe what we'll see is an evolution. As more stories come out, and they're going to, we're going to see the security of Face ID crumble. That's clearly going to happen. It's been about 10 days now, and it's already starting to happen.

Again, for convenience and for people who are like, oh, yeah, I don't really have anything on my phone that I care about, I get it that it offers that tradeoff. I would be reluctant to use it because, first of all, it's a brand new, unproven technology. It'd be crazy to immediately jump in only on Apple's assurances, which already seem to have been overblown. This to me seems substantially weaker than Touch ID because none of those instances of unlocking, opportunistic unlocking which we've seen would have occurred with Touch ID. There may be some fingerprint collision, but it just wouldn't be as obvious as this. So my feeling is...

Leo: But you're basing this not on the most recent attack, because that's unlikely. You're basing it on a presumption it's going to get easier.

Steve: No. If somebody had - there are pictures of me all over the 'Net. And so you could take those photos and reconstruct...

Leo: No, no, they didn't use photos, remember. They did a face scan. They needed a 3D scan, which they did, took five minutes with a scanning device. You can't use a 2D - you can't do this. This is an example. It's scary, but it wouldn't work.

Steve: But you didn't let me finish.

Leo: Okay.

Steve: Multiple 2D pictures can be used to create a 3D model. That's what I'm saying.

Leo: Okay.

Steve: If you have a picture of me from this angle and one from this angle, and there's all kinds of pictures, from that you can create a 3D model. That we know about. I mean, that's done all the time. So photos are inherently - they provide a huge amount of information. And I'll bet you that what we will see is somebody taking a bunch of 2D

photos to reconstruct a 3D model and then using it to spoof this.

Leo: You consider the fingerprint better? Because you've been using that.

Steve: Yeah, I have been.

Leo: Much easier to steal your fingerprint.

Steve: Okay.

Leo: Isn't it? You disagree?

Steve: My fingerprints are not all over the Internet, not all over the public Internet. So somebody could present me with a 3D model of my head, having never encountered me before.

Leo: Right.

Steve: Because of...

Leo: [Crosstalk] your phone, so it's not like they could do it remotely over the Internet.

Steve: Correct.

Leo: They're going to come to you at some point. So they could just follow you into Starbucks, get your fingerprint from a Starbucks mug, and then do it. They're going to be proximate to you eventually.

Steve: Yes, true. Yeah, and so I think my greater point is that my phone is, I mean, look at all of...

Leo: Well, I understand that, I mean, yes.

Steve: Look at all of the attention Apple has given to the security of their device.

Leo: Sure. Get my phone, you've got me. I agree with you on that.

Steve: Yes. They have thrown that out the window in the name of convenience, which is

unfortunate. I may, I mean, if it were possible to use Face ID to sort of get some surface-level access to the phone, and then need to authenticate with a substantial password, then maybe that would make sense. I may just not use it. It may not be a feature that I use. I could turn it off and not use it until we know more. To me, Apple's claims are not standing up. And it would be crazy to rely upon an untested, brand new feature like this until we know more.

Leo: That makes sense. I think that's reasonable, yeah.

Steve: Yeah. So it'll be interesting. And I can't wait to get mine because it does, I mean, in terms of processing power and the better screen and just, you know, I'm several generations behind. I'm back on the 6s. I didn't go to the Plus, nor to the 7 or the 8. So I've been holding back because you know me, I'm still using Windows XP.

Leo: Is part of this because you presume yourself to be a target? Or would any average Joe on the street have this concern?

Steve: Our listeners would have this concern if they have LastPass and Authenticator. I mean, the problem is, at this point, if someone gets into my phone, they could take down my life. They could take GRC off the 'Net. I mean, they could, if they had access to my authenticator and my email, because email comes into the phone, email of course is everyone's last resort account recovery. So they could point GRC.com to some other IP, to other name servers, and I would have a hard time getting it back.

Leo: I would submit the real risk is that our phones now contain so much stuff. And I would also say that I don't think Touch ID is necessarily any better than Face ID. It's more mature, but I don't know if it's any better than Face ID. In fact, I think it probably isn't as good as Face ID. It's a lot easier to make a fake fingerprint than it is a fake face. So, I mean, if you've been trusting Touch ID, you know, somebody has your phone, they probably have your fingerprint.

Steve: Yeah.

Leo: And we know you can easily spoof Touch ID with a fake fingerprint. So, I mean, I would agree with you on two points. One is that your phone contains your life, and that needs to be secured better. But the other, that this is a new technology, and we don't know how bad it is yet. We knew that as soon - I mean, I knew anyway, as soon as the iPhone X came out, that we'd start to see these articles, as happened with Touch ID.

Steve: Yeah. Yeah. I guess it's the ease of it happening in instances with people's children or people's lookalike...

Leo: No one looks like you, Steve. You're safe.

Steve: Thank goodness for that.

Leo: Yeah.

Steve: Okay. So a security researcher came up with a clever use for NTFS file system links. NTFS, very much like Unix, has long had this concept of a symbolic link. There's something called "directory junctions." There's hard links, there's soft links, and so forth.

What this researcher recognized was that, when AV systems detect malware and quarantine the malware in another directory, and when the user has the operating system privileges required to say, "Oh, no, that's not malware, that's a false positive, bring this out of seclusion," just this combination of features, along with something we've talked about a long time ago which is the DLL search path that exists in Windows, by cleverly combining these things, it turns out that this researcher was able to induce most of the industry's AV manufacturers to immediately revise their products. Trend Micro, Emsisoft, Kaspersky Lab, Malwarebytes, IKARUS, and ZoneAlarm have all been contacted and have released fixes. Some others who have also been contacted have not yet released fixes.

So here's the ID is when an AV product quarantines something that it considers malicious, it moves it into a quarantine directory for safety, to get it out of the normal operating system location, the download directory of wherever the user may have had access to it in order to sequester it. However, by the use of this NTFS file system feature known as "directory junctions," it is possible to place the quarantined directory into the Windows system DLL search path. The search path is a series of directories through which Windows will look, will automatically look in order to find a DLL which an application is requesting. By default, the system looks in the same directory as the application itself.

And I should mention, I'm sure most of our listeners know that a DLL stands for Dynamic Link Library. It was intended in the original design of Windows to be a way for applications to share libraries of functions without having to include them all in themselves. It turns out to have really pretty much collapsed because of conflicts among DLL versions. Microsoft has almost completely given up on the idea.

So now there's this new WinSXS, Windows Side-by-Side, technology which attempts to basically give up on this notion of sharing code completely because it ended up causing more trouble than it was worth. No longer do we really need to worry about hard drive space and economizing on the duplicate presence of common libraries and so forth. So once upon a time it worked. Now, not so much. However, all of the technology for backwards compatibility's sake still is there.

So there have been in the past exploits which relied upon Windows search order to get a malicious DLL upstream in the search sequence of where the valid one was actually residing in the file system so that Windows would encounter a same-named DLL and instantiate it, execute it, essentially. And in a DLL there is essentially an on-load function which is always called when the DLL is loaded into memory. So if you combine an AV program's decision to sequester something with the ability to essentially map the sequestered directory into the DLL search path, and if the bad guy then names the sequestered file something that any other program may ask to have loaded into its own process space, all of those things together actually do create a reliably and robustly exploitable hole which a huge number of AV is currently exposed to.

It's being fixed immediately. It's known as AVGator, is what this guy named it, #AVGator. And it's significant that it can only be exploited if the user's account is allowed to restore previously quarantined files. So one of the things that is probably a mitigation in enterprise environments is that normal users would be blocked from restoring threats that their AV had decided to quarantine. Hopefully that's the default policy, as it should be, in which case this would not be a problem, assuming that that's the case, that regular users cannot deal with this themselves and pull things out of quarantine.

The only real resolution for this, because this essentially isn't something easily turned off, is to make sure that you keep your AV up to date because we don't have full disclosure yet from those companies which have not patched, but any that are exposed have probably been notified and are in the process of fixing this very clever discovery about how to take advantage of a set of longstanding features in Windows.

We've talked a lot about Vault 7, which is the continuous dump of CIA material, programs, tools and so forth, from WikiLeaks. They've now started what they're calling "Vault 8," which is the release of the source code for a system known as Hive, which appears to be the CIA, the U.S. Central Intelligence Agency's malware or spyware control system. So they're leaking this under the name of Vault 8, and we're learning a lot about how this works. Okay. So this Hive, as it's known, presents itself as a network of publicly available web servers. So, for example, you would have a VPS, a Virtual Private Server, hosted on some random hosting site. The domain would be registered and, from all external appearances, looks just like a website.

If you go to that domain with your browser, you see some website, which is a front, essentially, as a control point for any of a number of implants which exist out in the world on machines that have access to the public Internet. There is a technology known as "optional authentication." Back in the old days, and I'm sure you remember this, Leo, you could go to a website that would suddenly hit you with a pop-up that was requesting your username and password. That's always been built into HTTP as mandatory authentication. That is, going to this page, you would be forced to provide a username and password in order to proceed.

Well, there is optional authentication where you're not stopped, I mean, you're not prompted at all, you're not stopped, but you can provide authentication as part of the query headers to the site. In which case the server will see that you are authenticated and be able to treat you differently; and, if you're not, you'll still have access. So this is a clever means, essentially, of hiding access in plain sight. And there are any number of these servers can be out in the world, just looking like a normal, legitimate website. People go there. You see text and pictures. They could be blogging sites. I mean, they could be anything. But only if an implant knows how to add its own authentication to its queries will it then go to a different place.

So the front is these VPSes, these Virtual Private Servers. They then establish a VPN connection to one or more proxy VPNs back somewhere else on the Internet which then, depending upon whether the query has this optional authentication or not, if it doesn't, routes that request to a so-called "cover server" to just provide cover to make it look like, oh, this is just a regular website. Otherwise, it goes to something known as the "honeycomb server," which then provides essentially the command and control to these remote, publicly located implants.

So certainly we've talked often about command-and-control systems for botnets and what this next generation of leaks, this Vault 8, which is leaking the source code for this technology, and which security researchers are now crawling all over, reveals the structure that, if we are to believe this, that our U.S. Central Intelligence Agency has

designed in order to allow things that are out in the wild to essentially phone home in a way that isn't obvious.

The other characteristic of this is anyone looking at the traffic would just think, oh, look, you know, this service is for some reason accessing a website. And if you saw it doing so, and you typed that domain name into your own browser because you were curious, you would get a regular-looking domain. So it's clever. Not high tech, uses longstanding technologies, and there's no reason to believe this isn't what they're doing. It certainly would make sense for them to be doing exactly that.

Leo: Given the evidence, not conclusive, but the evidence that WikiLeaks is a mouthpiece for the Russian government, I wonder how credible it is in leaks like this, of this sort? I mean, obviously somebody made this. This isn't something somebody at WikiLeaks, you know, Julian Assange didn't write this in his spare time.

Steve: Right, yeah, right. There's a serious body of source code here.

Leo: But who knows what its real provenance is; right?

Steve: Yeah.

Leo: I mean, can you definitely say this is from the CIA?

Steve: No.

Leo: Not the FSB, for instance?

Steve: I guess the only way, I mean, if someone were really interested, you'd have to find one of these. You'd have to look at the credentials and then track down the VPN and, like, do some forensics. But, yeah, I mean, as we know, attribution is always the problem with these things. So unless the CIA said, "Yeah, you got us," you know...

Leo: Unlikely.

Steve: Yeah, exactly.

Leo: How about Vault 7? I mean, I think that's accepted now that that was fully the NSA's stuff; right?

Steve: Yeah.

Leo: Yeah. So that doesn't mean, I guess, that the information WikiLeaks has is false. You might question their motivations, I guess.

Steve: Yeah. And if nothing else, it's interesting, and it certainly seems feasible. But again, you're right, Leo, we don't know one way or the other. I mean, this is leakage that wasn't intended to be leaked. So, I mean, well, leakage that could have any source.

Leo: Right.

Steve: So we had a mass shooting Sunday before last at a church in Texas. And naturally, immediately afterwards, much as happened with San Bernardino, the FBI wants to understand everything they can about the background of this person who we pretty much know did the shooting as a consequence of the evidence trail. And what reporting, and this is from Reuters reporting, Apple claims that it immediately contacted the FBI to offer their assistance in unlocking this Texas shooter's iPhone. The FBI is arguing that this is yet another case of them not having critically required access to encrypted data.

The problem in this case is that there was an immediate clock that was ticking. We know who the suspected shooter was. And if the shooter had been using their fingerprint to unlock their phone, and if even post mortem this fingerprint had been applied to the phone, presumably a thumb, within 48 hours - after which we know that Touch ID stops working, and you have to be using your passcode - then there was this window of opportunity. And so this has caused some controversy because the FBI did not take advantage, the reporting alleges, of Apple's immediate offer to provide any assistance that they could. We don't know what the FBI did with the phone, whether they in fact got access to it or didn't.

Tim Cushing, writing for Techdirt, looking at all of the coverage, wrote that: "If everything alleged by the Reuters report is true, the FBI's struggles are of its own making." But then again, there is still a lot of information that is unknown.

Leo: Now, I actually thought this was weird. Apparently Apple said, "Well, you know, you should have called us because you could have just put the dead guy's finger on the phone and unlocked it." What?

Steve: Right.

Leo: Do you think Apple really said that to them? They said that the - remember the San Bernardino thing, they said: "You should have called us because we could have, you know, there was a chance he was synced with iCloud, and we could have given it to you then."

Steve: Exactly, yes.

Leo: That I believe.

Steve: Yes, yes.

Leo: But can you put the dead person's finger on the phone and unlock it?

Steve: Heck, yes.

Leo: But doesn't it have some sort of infrared to detect heat? I thought it did. Maybe I'm...

Steve: There's not IR because it's capacitive.

Leo: I thought that you had to use like a sausage - oh, capacitive would still work with a dead finger, I guess.

Steve: Yes.

Leo: But you've got to do it right away because 48 hours afterwards [crosstalk].

Steve: The gummi bear fingers also work.

Leo: That works, too, okay, all right.

Steve: Yeah. So it has to have ridges in order to function capacitively. And of course then Dianne Feinstein immediately dusted off her legislation which had died for lack of support near the end of the previous U.S. administration, when nobody was jumping to move on this. So we're still in the middle of this struggle. And so here's yet another event where the FBI is using this instance to complain about their inability, in a very high-profile event, their inability to get access to a criminal's encrypted data; Apple trying to defuse this saying, "Hey, we offered. We stepped up immediately." They probably did, from a PR standpoint, just saying look, we'll provide any help that you need of any kind because, as you said, Leo, arguably there were some mistakes made immediately in the San Bernardino case which may have, if things had been handled differently, may have allowed the data to be recovered.

And I'll just remind everyone that, as we've said on this podcast, there is a solution that Apple doesn't want to be forced to provide, which isn't a back door or a front door or a golden key or a side door or an unlocked door, the idea being that Apple is maintaining a connection to every one of their iDevices. I mean, that's part of what they're doing. These things are connected, and they're getting updates, and they're doing iClouds, and all of this is going on.

Ultimately, in every single one of these devices is a master 256-bit symmetric key which is the thing which unlocks the stored data. And we know that it is super well guarded by multiple layers of security, by hardware, by enclave processors and all this, and that Apple doesn't have it currently. We believe them that the device, when it initializes the first time, it uses available entropy to invent this key for its own use, and it never leaves the device. That one thing is all that would need to change. That is, when the device invents this key, comes up with its super-secret, super-guarded, 256-bit symmetric key, that key could be encrypted with a very long, very safe, master transport key, that is, that's in the device.

This would be a public key using public key technology, which could be used to encrypt this 256-bit per-device symmetric key for transport to Apple, so that Apple then has this unique per-device key in their database. And if called upon on an individual per-device basis, Apple could decrypt that using the highly guarded at Apple matching private key, which would allow per-device access to the contents of the phone. They don't want to do it. I get that. I respect that.

But there are solutions, is my point, to providing under law single-device access which Apple could at their discretion with this technology choose to make available if legislation is enacted to cause that to happen, without anybody getting a master, any third party, any law enforcement getting any sort of carte blanche access to all of the iPhones in the world.

Leo: The problem with that, of course, is that they're afraid, not so much about the U.S. government, but other governments asking for that kind of access.

Steve: Ah, right.

Leo: Which they would have to provide.

Steve: If it exists, then, right.

Leo: So acknowledging that that exists would be a problem.

Steve: Yeah. So yet another way - we keep coming up with and discussing bizarre ways of exfiltrating data. We've had hard drive noise. We've had ultrasonic sound coming from the speaker. We've got flashing lights on our routers. Some of them have been debunked, some not. I mean, even the classic RF emanations from CRTs, and then LCDs, and reading contents of screens through the wall just by picking up leakage of information. Side channel attacks, figuring out what key is being decrypted on the processor when you share it with a VM, I mean, there's just all kinds of stuff.

And there's a new one in town. Imagine some malware in a mobile device which, during a video conference, puts what looks like just sort of a band of static on the bottom of your screen. And you kind of look at that, and it's like, what? This is weird. It looks like in the old days when you didn't have your vertical alignment adjusted on your TV, and you were seeing noise off to the side.

And so the point is, it turns out that, although a little bit frustrated by video compression,

it certainly is possible - and it has been done now, there's been a proof of concept of this - where some malware added some noise, some static to the bottom of a video transmission which, at the recipient end, allowed that what looks just like static is actually fast-moving data in order to exfiltrate data across a video link, where file transfer had been explicitly denied.

And so anyway, just yet one more means of exfiltrating data from a system. Pretty much, I mean, you couldn't use traditional steganometry because compression would completely destroy the 24-bit color. And as we know, it makes the images blurry. It reduces the color depth. So it is a challenge to do this. But it certainly is possible from a technology standpoint to pull it off. And a group at PenTestPartners.com, they made it happen and demonstrated more than proof of concept. You actually could do it just by putting something that most users would think, oh, that's weird, some static on the screen. I don't know what that's about, but who cares?

So Google is moving forward with Chrome security and will in the next couple months be adding three new features to Chrome to fight malvertising.

Leo: Oh, boy.

Steve: And these are good. In the first case I'm glad that Google is willing to do this because it's deliberately breaking some proper behavior, but necessarily. So, okay. So with release 64 of Chrome, which is slated for January of next year, January 2018, Chrome will be blocking script-driven iframe redirects. This is a consequence of the history of abuse by embedded malvertising. Chrome will no longer accept URL redirections triggered by JavaScript code residing within iframes.

So, I mean, iframes have been sort of a controversial problem on the web for a long time. They are a means for one web page to embed another web page within itself, literally on the real estate of the hosting web page. In the HTML you declare a frame, a horizontal by vertical frame of a certain size or percentage of the page. And in this declaration you give it the URL of the web page to fill the frame with. And it doesn't have to be the same domain.

So this is inherently a cross-domain thing. This is a means of embedding any other web page. And in fact it's been used controversially by some less reputable sites to essentially embed good sites, like steal site content from some other web server. It's like, oh, look at our stuff, when in fact all they have is an iframe pointing somewhere else, but people get to that somewhere else by going to the primary site. So it's a convenience. It was the sort of thing that someone said, "Hey, wouldn't this be cool," back in the beginning, the dawn of the web. And everyone said, "Yeah, okay, fine, we'll put that in."

So the point is that this day and age it's typically the mechanism used for advertising. So you have a web page, and you want to host third-party ads on your page. So you give them an agreed-upon rectangular area on your page, and you point that to the web server's URL. The web server, when queried, they're going to see from the referrer header where the query is coming from. Oh, it's from these guys who are hosting our ad. And of course they also get cookies that belong to their domain, so they know who you are, looking at the site that is hosting the third-party service. So they get all this information; and they return, hopefully, an ad which you care about and which the site receives some remuneration for.

The problem is this very powerful facility can also run, I mean, it's a full web page, which

means it can run JavaScript. It can run code. And this has been used by malware that is injected into legitimate advertising services in order to get users to click on something, and they click on something, and that installs something into their computer that they don't want, and it's all bad. So the point is this is all standards-based. And so I take my hat off to Chrome for deciding we're going to break something which is being abused because, even though it's legitimate, that is, it's not something that needs to be fixed, it's always been possible. They're going to say, eh, we're going to stop making that possible. We're willing to take the hit for script running in an iframe that attempts to redirect to somewhere else. We're going to have Chrome look at that behavior and not abide it, not follow that redirect.

So I'm glad that Google is willing to do this. They sort of have to be the first people to be the icebreakers, after which other browsers could follow, presuming that this doesn't cause too big a problem. And they must have analyzed this in order to decide that, yup, the benefit for the user outweighs the tiny loss of functionality that this could incur. I mean, and especially since Google is largely advertising supported, and this is something that maybe a legitimate advertiser could use. They're just going to have to stop doing that because, after January and release 64, Chrome won't follow script-driven redirections from an iframe.

Then two months later, in March of '18, Chrome will begin blocking what's known as "tab under" behavior. "Tab under" is the term for a web page, sort of a clever scheme by which a web page can bypass Chrome's built-in pop-up blocker. What the page does is opening links in new tabs and then redirecting the original tab to a new URL. So starting with Chrome 65, obviously the one after 64, which we were just talking about, with the next iteration of Chrome targeted for, I think it was early March 2018, that redirection attempt for the original tab will be blocked, and Chrome instead will display a notice at the bottom of the page, just letting people know that some JavaScript's attempt to redirect this page to a different URL has been blocked.

And finally, there are pages which abuse the user's expectation of what will happen when they click on something. This is important because there is a huge difference in the handling of things that JavaScript can automate without user interaction. That's why malvertising wants you to get to click on something. It's why clicking on an ad can be dangerous, because browsers handle passive and active differently. Well, it turns out there are suspicious and malicious sites which will, for example, put up a video that the user doesn't want; but the Pause button, which we would tend to click on in order to say "Stop this playing, this is loud and obnoxious," the Pause button is actually an href click that does something other than pause.

Leo: Ooh, that's sneaky.

Steve: Yes. It's also possible to put up a transparent overlay on the page so the user, in clicking on something, is actually clicking on a transparent surface which receives the click, instead of what is visually going to receive the click. That gives scripting or the browser much higher level of permission. In fact, we use this characteristic in SQRL. When you click on the QR code because you want to do same-machine logon, that initiates an href jump to the SQRL client running on the system as localhost. The scripting can't do that. That would be cross-site. That would be going from the site you're logging into over to the localhost domain or IP on your own machine. Script can't do it, but a user-initiated click can.

So that's a perfect example of how permissive it is. And of course this has to happen

because that's the way the web is linked together, right, is URLs linking you to other domains. So if the user clicks on something, the browser goes, even if it's a jump offsite, somewhere else. And so what's happening is the problem is this is also - maybe it's benign. Many times it could be a funny-looking button saying, hey, if you want to go to the Disney site, click here. So that's legitimate. So there isn't technology that Google can deploy to block that because there may be very valid reasons for having a click take you somewhere else.

So the way Google is going to handle this is they have something called the "Abusive Experiences Report." And it's going to take the form of a blacklist of sites known to be using misleading UI elements, misleading user interface elements such as Play or Pause buttons or whatever. And so the idea is that, as I understand this, website owners can register their site with Google to receive early warnings if Google believes their site is using these types of misleading UI elements.

So this is the way Google's decided to handle it. They don't want to just blacklist sites without warning, but they're going to blacklist sites that perpetrate this kind of abuse. So they've decided what they'll do is they'll create a mechanism - and there's no, you know, not every site in the world has to do this. But they're saying, if you're worried you might get on this blacklist, you can register yourself with us in advance, and you'll be able to check to see if you are being blacklisted in this fashion and then work with Google or change the behavior of your site in order to get pulled off of this pending blacklist.

So I can't think of - oh, and these sites will then end up getting blocked by Google's existing built-in pop-up blocker. They will become blacklisted, and those UI functions which are what causes Chrome to worry about this in the first place, those will end up just not functioning. So things that Google believed were misleading will end up being turned off. So this is pretty aggressive; and, I mean, props to Google for being willing to do this because it's going to break some things, but there's no - I can't see a technological way to solve this problem because the issue is its misleading UI elements.

Well, you can't define that in code, so it's got to be Google learns that there's a certain site which is doing this, and so they blacklist that functionality on that site. Doesn't kill the site. You can still get there. But when you click something that people were complaining about, it just won't work any longer until the site fixes it. So again, aggressive, but I'm afraid that's what we've come down because the abuse is such that you have to take this kind of action.

Okay, Leo?

Leo: Yes? You rang?

Steve: You're not going to believe this one.

Leo: Uh-oh.

Steve: I know you have heard the term JTAG, the JTAG port or a JTAG connection. It's been around since 1985. JTAG itself stands for the Joint Test Action Group. But it's most known by anybody who's been playing with embedded stuff, like hard drives have a JTAG interface. In fact, Leo, you've been changing firmware on some of your devices using JTAG.

Leo: Yeah? I didn't know that.

Steve: That's what it is. It's a low wire count, sometimes two-wire, sometimes three or four, which is a universal standard. So any embedded processor has a JTAG interface. All the ARM chips and the TI chips, all of, I mean, the PIC chips, all these embedded processors, they have a JTAG interface. And it is incredibly powerful.

With a JTAG interface, which typically has a clock and a mode and a data in and a data out, it is possible to halt the processor, to read out the contents of its registers, to change the contents of its registers, including the program counter, allowing you to cause it to start executing from somewhere else. You can single-step it so you can say execute one instruction. Then you can again read out all the contents of its registers. In other words, it is a very powerful debugging interface.

You can attach this JTAG programmer, as they're called, clip it on the back of the chip, or find these JTAG programming pins on a device and essentially completely take it over remotely. When people have hacked hard drives or sucked the firmware out of a hard drive, it's the JTAG interface that allowed them to do that. And you're able to change the contents. You're able to read and write RAM, read and write ROM, suck out the firmware.

Now, it's possible to blow a fuse, as it's termed, after using this JTAG interface to do programming. And after you verify it's what you want, you're then able to irreversibly blow a JTAG programming fuse that turns off either some or all of the functionality in order to prevent, for example, your competition from reading out the contents of your firmware in your proprietary device. So that can be done. But believe it or not, a group known as Positive Technologies will be providing full information next month, but they have determined and revealed that, starting with Skylake and all subsequent chips, so Intel's Skylake chipset and subsequently, Intel - it's hard for me to even say this - has added an external JTAG interface to the USB ports.

Leo: Well, it's convenience.

Steve: Oh, my lord.

Leo: Wow. That's kind of amazing.

Steve: Oh, it's just - it's breathtaking. They call it the "god-mode hack" because it is. Now, it's not remote. So, I mean, thank goodness. But it means that somebody having physical proximate access to a USB port on a motherboard from Skylake onward is able to essentially do anything they want. This is complete access to the Intel Management Engine through the motherboard's USB ports. Intel has an acronym. They call it DCI, Direct Connect Interface. How convenient.

Leo: I take it they didn't blow the fuse on it.

Steve: Apparently not because these guys are going to demonstrate next month that it is possible to run unsigned code on the platform controller hub.

Leo: Just plug in a USB.

Steve: Any, yes, any given motherboard from Skylake and after. And essentially...

Leo: Is there not a fuse? Maybe, I mean, can I retroactively blow the fuse?

Steve: The problem is that would disable what Intel apparently thinks is for some reason useful. So it's just beyond me. Hopefully we will hear a statement from Intel about how they explain this. But, I mean, the JTAG port, it is god. I mean, it is a full debugging interface. And they said, oh, well. Remember last week we were talking about what was required. You had to go find the little BIOS chip, eight pins on the motherboard, and get out your soldering iron and your microscope and attach stuff. No. Turns out just plug in a sufficiently configured USB device, and you're good to go.

Leo: Wow.

Steve: Making the world much easier for the rest of us.

Leo: I'm going to pour some epoxy in my USB ports.

Steve: Wow. Yes.

Leo: I'm thinking, I'm just thinking some of the usefulness would be in manufacture. If you update your BIOS on your motherboard, and you fry it, that would be one way they could get in and replace the firmware and the motherboard, I would guess; right?

Steve: Yeah. I just, I mean, you could have a dedicated JTAG port. Why export it over USB?

Leo: Yeah, that makes it pretty easy and convenient. It's easy to update.

Steve: Now, who are they saying "yes, sir" to is what comes to mind. I mean, yes, not a remote hack. But, boy, it does mean that drive-by system, deep system compromise becomes possible.

Leo: You've got a little device, you plug it in the USB port, press a couple of buttons, bzzz bzzz bzzz, walk away, and you now own that system; right?

Steve: Yup. Yup. You've just changed the core, as it's known, Ring -3. You've got...

Leo: It's that low. It's so low that you couldn't - and of course antivirus wouldn't, I mean, talk about rootkit, antivirus wouldn't see it because it's in the firmware.

Steve: Yes. That is, it is the processor which is executed before the main processor starts. And we've already seen that it's possible for that code to be injected into the address space of the higher level Intel processor. So, yeah, it's a complete sub-rootkit capability.

Leo: Wow.

Steve: Amazing.

Leo: I wonder what Intel's response is to this.

Steve: That's going to be interesting.

Leo: They need to take this Management Engine out of all future chips. That's ridiculous.

Steve: Yeah, it's really, I mean, and I'm sure we've talked about how Google is scrambling around trying to get it out of their servers in all their server farms because it just...

Leo: Oh, it's in everything.

Steve: ...terrifies them, yeah.

Leo: Is there a physical hardware way to disable it? There's no trace. It's in the die; right?

Steve: But the problem is, if this is a JTAG interface, this overrides. There is a bit, that bit that the NSA uses because the NSA doesn't want this in their hardware, either.

Leo: No.

Steve: So there is a bit...

Leo: They want it in our hardware, not their hardware.

Steve: Exactly. There's a bit that they can set which causes only those two critical modules, those boot config modules, to just - because you need this thing to set the clock frequencies and set the wait states on the DRAM. It's that our motherboards are so, the BIOSes are so capable now, where you're able to change, like, wait states and clock speeds and everything, it's just amazing.

Well, all of that flexibility is this IME. That's the processor running this MINIX OS which allows all that to happen. Because that has to happen underneath the main Intel CPU in order for it to even get up and going. So what this bit does that the NSA can set is it causes only that to happen. And then all those other modules, for example that module that was causing me to, well, I was going to say "pull out my hair," but no, apparently I did that already.

Leo: Because you had a vPRO Ethernet card.

Steve: Because I had, yeah, actually it was an older motherboard that was having some sort of ARP storm caused entirely by the fact that this Intel Management Engine was doing its own thing on the port I was using. All I had to do was switch to the secondary port.

Leo: Because that port was the vPRO.

Steve: Which the IME is - yeah.

Leo: That was the managed Ethernet port. Now, previous hacks of the IME have required that you have it be enabled and it be a vPRO machine. But it sounds like this IME is in every chip Intel makes; right?

Steve: Yes. It is the thing...

Leo: There's just no interface to it on non-vPRO systems.

Steve: Right.

Leo: But there's still a JTAG interface. Or no?

Steve: Well, it's the same chipset. It's the chipset.

Leo: It must be; right.

Steve: Yes. Wow. And, I mean, I'm sure everywhere is like, at the moment we get more information, this makes the hack we talked about last week obsolete immediately.

Leo: Right, right, right.

Steve: No.

Leo: I don't need that thing.

Steve: You don't even need to open your box.

Leo: I don't need to clip on.

Steve: Exactly. There will be how-to's, like how to turn off your IME forever. Plug this in, and it's going to be shut down.

Leo: Well, that's what somebody should make is a USB key you plug in that disables IME.

Steve: That absolutely will happen.

Leo: Oh, I hope so.

Steve: We can predict that now.

Leo: They could sell that for a hundred bucks. I would buy it right away.

Steve: Yup. Okay. So we've talked - there is a new jargon, a new term of art, "cryptojacking," which I think is wonderful. That's, of course, hijacking your power and CPU for crypto mining for which you gain no benefit: cryptojacking. An analysis by a researcher, Willem de Groot, he took a look across the Internet, found 2,496 sites, so four fewer than 2,500 sites, doing crypto mining whenever users visited them. And this of course was - they were all using the popular Coinhive Monero cryptocurrency mining JavaScript.

But it turns out that the beneficiaries of that had a very interesting distribution. In the script, there for anyone to see who does a "view source," is the account, the Coinhive account to which the mining effort is credited. 85% of these just four shy of 2,500 sites, 85% of them were benefiting just two mining accounts. Which, whoa, is not what you expect. If crypto mining was for the benefit of the hosting site, then you would expect 2,500 different Coinhive accounts, one per site. No. 85% of them were linked to just two mining accounts; while the remaining 15% were spread out over unique Coinhive accounts, although those 15% had ID tags which, while different, all had the same format and pattern involving the domain of the site that was hosting them, meaning that a third group was using a different approach. But one group accounts for the remaining 15%.

In other words, of the nearly 2,500 sites that Willem de Groot examined, there were three, a total of just three individuals or groups behind all of this cryptojacking, and in every case he found other malware that was present on the servers. So the conclusion here is that these sites are compromised and compromisable, that they were already hosting and probably known to be vulnerable to various sorts of website compromise. And so the bad guys thought, well, we've already got some of our malware on these sites. We might as well do a little mining for ourselves while people are visiting these sites.

So essentially three different groups are responsible for what appears to be an explosion in cryptojacking. It's not actually all these individual domains hoping to make some money from their visitors. It's bad guys, three of them in total, who have used compromised features of those websites to insert cryptocurrency mining. Now, what this says to me is that Coinhive, which is the single focal point of all this, should clearly be shut down. I mean, should shut down itself. These guys should say, okay, we had an idea, it works, but it's being abused.

The problem is that it's sort of a Hobbesian bargain here because remember, when we first talked about it, I was shocked by the high percentage that Coinhive retained of the earnings. So they're making a lot of money on the fact that, I mean, they're essentially complicit in this. They must know what's going on. They know who's hosting their script because those queries are coming to their servers to load their script into innocent users' browsers. The problem is they're making a lot of money. Well, some money. It's not clear that any of this actually makes a lot of money because I don't think these sites are very popular. But they're obtaining a substantial portion of the proceeds. So they've essentially become agents of these malicious hackers and, I would argue, probably willful agents.

In terms of what to do, as we've talked about before, things like uBlock Origin are already blocking this. We know that adblockers are blocking it. But I did want to mention another old trick of the trade, and that is that the hosts files is 100% effective in blocking this. There are two domains from which all of this is hosted: Coinhive.com and Coin-hive.com, coin dash H-I-V-E dot com. If you were to just go to your browser, as we did back when we first talked about this, and put in Coinhive.com, up would pop their site.

Well, you simply use the hosts file to redirect Coin-hive.com and Coinhive.com to the localhost, 127.0.0.1. And the hosts file is still the first place that Windows goes before it performs a DNS query. And that will prevent any site you go to from providing script which would cause your browser to look up the IP for Coinhive.com. It'll do that. It'll get 127.0.0.1, where almost certainly you have no web server running; or, even if you do, it's not Coinhive.com, and you will not get any scripting running. So I like that as a tip for anyone, our more technical users who like the idea of just preventing their system or any browser on their computer, whether it's got uBlock Origin or an adblocker or not, from inadvertently running anything from Coinhive.com.

And when I first heard that JavaScript was mining bitcoin, my first reaction was, whoa, how can that be fast enough? Because maybe 10 years ago; but, boy, how could it be feasible now? Well, it turns out that it is 100% the fact or fault of a technology we first covered 2.5 years ago, in April of 2015, known as WebAssembly. We mentioned this back then, that work on something known as WebAssembly had started, and that browser makers were going to join forces to create a binary bytecode universal, essentially, assembly language, like machine language, a virtual machine for web browser-based applications.

And as of last month, October, when we got iOS 11 and thus Safari and the Fall Creators

update for Windows 10, where Microsoft's Edge HTML 16 occurred, now all major browsers support the WebAssembly standard. Firefox and Chrome were first this past summer. And then the other Chromium-based browsers, like Opera and Vivaldi, they inherited this feature as soon as it went into the Chromium stable version. And now we've got Safari and Edge. And I don't know about IE11. I don't know whether that got it or not.

So as it is today, JavaScript, well, without WebAssembly, as we know, JavaScript source code is provided by a web server in ASCII, loaded into the browser, and must then be essentially interpreted. And typically it's compiled down into JavaScript bytecode by a JIT, a so-called Just In Time compiler, which compiles it sort of as it goes in order not to have a huge compilation burden upfront. People want things to start immediately. So the idea is that the first time it encounters not-yet-compiled JavaScript, it compiles it incrementally. And so, if there are areas that aren't being used of JavaScript, it just doesn't bother with those. But this takes time. It wastes resources because, after all, every time you go back, you're getting the same script again and having to recompile it again.

So what we've done is the industry - and the reason I'm bringing it up is not to complain that it's being used by Coinhive, but to mention that this demonstrates the incredible performance. The browser-based gaming guys were all over this immediately because it offers stunning performance. It's about a factor of 20 times faster than JavaScript for parsing the language because what it does is it means that this is precompiled.

So the person who is authoring this for a site compiles the JavaScript or the C, C++, or Rust, because there are multiple compilers now, into this common bytecode. That then is offered for download to the browser. It comes in, and it can be executed instantaneously. No need to interpret it. It is far smaller because it is not ASCII, it is binary. It's a binary blob of bytecode, which the browser can immediately start executing and execute faster. All the optimization can be done statically once, and then everyone who downloads it gets the benefit of it. You don't need to use JavaScript.

Just for example, people who want to write in C or C++ have had to do something known as "transpiling," essentially transcompiling, from another language over into JavaScript in order to be able to offer it to browsers. Now you can have different frontend compilers that will compile into this common WebAssembly in order for everyone to have access to it. So we're actually there. And people are a little surprised that this happened without endless committees arguing about this or that feature. I mean, it is established. It is version proof. It's been, you know, it just sort of happened without us really seeing it happen.

If anyone is interested in more, WebAssembly.org is the site, W-E-B-A-S-S-E-M-B-L-Y dot org. And it's, as they say, it's in there. It's already in our browsers. So, very cool. I think that, as we begin to see it adopted, certainly companies like Google will jump on this immediately. And if people are still using pre-WebAssembly browsers, that would be visible in the user-agent field of the query for the page. So a server could say, whoops, this guy's using a browser that is not WebAssembly-capable; so, yes, we'll feed him the JavaScript equivalent. But that will happen less and less as those browsers die off and stop being used.

And eventually, oh, there's one more thing I forgot. This is another beautiful part of it. The problem was, wait a minute, I'm downloading a big binary blob for which I have no visibility. Because it's very nice to be able to look at the JavaScript which your browser has pulled down. It turns out they thought of that, too. There is a very clean text version of the binary. So you'll be able to do a view source on a page containing a WebAssembly

invocation, a WebAssembly binary blob, and view it in textual form in a meaningful way. So, I mean, it's not going to be - probably it's going to be like assembly language. But still, there is a viewer for this as part of the offering and as part of the support that the browsers offer. So, very cool.

This is just a quickie. And Leo, you'll get a kick out of this. The site is Gcc.godbolt.org.

Leo: Love the name.

Steve: Gcc.godbolt.org. It is very cool. It is an interactive, side-by-side display of...

Leo: Oh, look at that.

Steve: ...GCC compiler input and output. And what's neat is you can, over on the right-hand side, you can change which compiler version and which target chip, like x86, x64, ARM and so forth. And in every case you see the output of the compiler, given the input. And over on the left, if you highlight different lines, you'll see that the lines are - the source input is in different colors. And so it shows which line expands into which instruction.

Leo: That is really neat. That is so neat.

Steve: Isn't that cool?

Leo: Yeah. This would be a good way to learn assembly.

Steve: Yes, actually, it would be because you're able to see. And if you put something more fancy in for the source code, the example they have is just a simple square of a number. But it's a good starter. But you put something else in, like a little bit of a program, and you can actually see the code that the compiler generates for all different chipset families.

Leo: Yeah. This is the ARM code. Wow.

Steve: Yeah.

Leo: This is so cool. That is so cool. I really like this.

Steve: Yup, Gcc.godbolt.org.

Leo: MIPS?

Steve: Yup, exactly, the MIPS, yeah.

Leo: PowerPC, wow. That is neat.

Steve: And just very, very cool to compare.

Leo: So it's an interactive C compiler. Oh, you could try different languages, too. You can write something in Swift or Haskell or Rust.

Steve: Ah, nice.

Leo: And then see how the Rust compiles or assembles out.

Steve: Yes.

Leo: Nice. What a nice little hack. Godbolt.

Steve: Yeah. Godbolt. Gcc.godbolt.org. And I did want to mention we've talked about the KRACK vulnerability and updating routers. I was looking at mine the other day, and there was a little yellow exclamation point flashing. And I thought, oh, what's that? And it's like, oh, there's an update for your firmware. In this case it was an Asus RT-AC68U router. And I was of course very curious to know what the firmware was doing.

And sure enough, first thing on the list, security fixed: Fixed KRACK vulnerability. Then it fixed a series of CVE numbers. Let's see: three DNS, two or three DHCP, predictable session tokens, logged user IP validation, a GUI authorization vulnerability, a cross-site scripting vulnerability in their own web interface. Added some features: HDD hibernation if you have a hard drive hooked onto it, a URL filter black-and-white list, bandwidth limiter on the guest network.

Anyway, so this is the kind of - and this is from Asus. This is the kind of support we are now recognizing we have to have for our connected devices, especially those that are facing the public Internet, as most routers are doing for us. So I just wanted to just - this was my real-world experience. It didn't do it for me, and that's the takeaway here. Yes, all this happy goodness was there. But it was waiting for me to log into the admin interface and notice that there was a little yellow exclamation point flashing in the upper...

Leo: Yeah, that's the only drawback. That's why you want pushed updates out; right?

Steve: Yup. And I think that - oh, and a lot of people mentioned the side-by-side firmware. We were talking about the emerging standard for updating IoT devices. And I should have mentioned, yes, that for example this has been done historically for Internet-connected devices. I mean, back from the Cisco Internet Operating System,

their IOS, there have always been multiple firmware images. And so you're able to update one, and then you switch over to it next time you boot. And I've got a bunch of Internet equipment that runs GRC, which is all multiple firmware, flashable firmware images, and then you choose which one you want to boot from. So that's well existing technology. But it's certainly, as far as we know, nothing like that exists for light bulbs. So the whole point is let's push this time-proven approach down into very inexpensive devices.

And I just wanted to mention also, I recently had to have a conversation with someone at Hover. And I was very impressed. Well, I had to do something that required a conversation.

Leo: We should mention they're a sponsor.

Steve: Oh, they are?

Leo: Oh, you didn't know Hover was a sponsor? Even better.

Steve: They're my domain name, yes.

Leo: Even better. Yeah, well, we had a Hover ad in iOS Today, and I used your name in vain because I know you're very happy.

Steve: Good. Yes, I am.

Leo: Yeah, they're a sponsor.

Steve: Okay. So, good, they're a sponsor. And, I mean, I fell in love with them before. As everyone knows, I was looking for a solution, an alternative to Network Solutions, who I had been with forever, have been with forever. In order to get what I needed done, I first received a PIN through email to my registered email account. I then had to have a telephone conversation where I had to read them back the PIN and feed them the current, because I have two-factor authentication set up on my Hover account, had to read them the instantaneously correct current six-digit two-factor authentication, which the guy I was talking to, Shane, immediately confirmed and said, "Okay, what do you need?" So that's what I want.

Leo: Nice.

Steve: I want that kind of frontline security for the people who are maintaining my domain name registration. So, yay. Also, a question that frequently comes up I wanted to touch on briefly. I saw this, Dave L. in San Rafael, so in your neck of the woods, Leo. The subject was "SpinRite on Hybrid Hard Drives," send to us on the 5th of November. He said: "My new laptop has a hybrid hard drive on which I have been afraid to run SpinRite in fear that the solid-state portion would be degraded by heavy reads and

writes. Can you advise listeners on the best way to run SpinRite on such a drive?"

And the answer is yes, of course, and there's nothing you need to do. From the very first days of SpinRite, SpinRite takes advantage of some features which have always existed from the beginning of the IDE world to disable all caching on drives. SpinRite has never wanted to test the RAM cache on drives which do caching. And so the hybrid drive is an extension of that. Basically it's a non-volatile cache where the things that the drive determines are read most often are kept in order to minimize read delays because it's possible to read from a big SSD cache, as we know, way more quickly than it is to move the heads and wait for the data to rotate around and get read if it doesn't exist in the much, much smaller RAM cache.

So essentially there's a hierarchy of caches: RAM, and then essentially non-volatile read cache, and then the big physical magnetic read/write memory. What SpinRite does is to disable all caching, and the hybrid caching is part of that. So that gets turned off in order that SpinRite has direct physical access to the magnetic media, which is what you really want to perform recovery on and testing on. So users need do nothing. SpinRite, thanks to the fact that caching exists and the hybrid sort of interstitial cache between RAM and magnetic surface is also a cache. That gets turned off, and SpinRite has direct access. So works just fine on hybrid drives.

Some quick closing-the-loop pieces. Christopher Ursich said: "The installer I downloaded today on November 11th from LastPass was signed on October 3rd, but with a cert that expired on October 29th. Is it good practice to trust or not?" Okay. So this is interesting. Code-signing certificates are handled differently than web-signing certificates, than certificates that websites use. There, because the nature is real-time downloading, like on a website, you are establishing a connection right now where you want to verify the identity right now of this website.

There the certificate the site is offering in real-time must have its dates be valid. We'd often talk about a site inadvertently offering an expired certificate. But the model for protection is different for code signing. With code signing, what you really want is to know that, at the time of the signing, the company doing the signing had a valid certificate. So what you want to prevent is somebody gets an old - some bad guy gets an old expired certificate and then uses that to sign their malware, which would then be trusted.

So the way this is handled is there's something known as a time server. And the code-signing system is able as part of what it does to, on the fly, query a time server. The URL for that can be provided by the user, or it's often contained in the certificate, that is, the certificate itself says where to query for time. So the signing process, when performing the signing, pings a time server, gets the current time, verifies that the certificate is valid, and binds all of this together so that what you get is you get the code with a valid certificate and the timestamp on which the code was signed and the assertion that, at the time it was signed, that time fell within the validity period of the certificate.

And when you think about it, that is exactly what you want. It makes sense because code that is going to be downloaded could sit on the shelf for years. And so 10 years from now you may download it. Well, the certificate is certainly going to have expired. So rather than, like, forcing all code everywhere to always be resigned, which is not practical, with a now-valid certificate, the idea is this use of a time server and a timestamp sidesteps that. It's what we really want. It verifies when it was signed it was valid. And then, of course, you check the signature to verify that it hasn't been changed since then. So great question, Christopher, and I'm glad you asked for clarification.

I love this. David Lemire says: "In the discussion of disabling Intel's ME" - the management engine that we've just been talking about - "in last week's episode," he says, "it created a terminology question." He says: "If killing a phone is 'bricking it,' is killing a motherboard 'planking'?" Which I thought was kind of fun and clever. Of course we call it "bricking the phone" because then you're just sort of holding a brick in your hand. In the case of a motherboard, well, what would a dead motherboard be? I guess maybe a plank.

Oh, and Marc Boorshtein said: "I'm not sure favicon.ico is a good login test." He said: "Usually anonymous since it's displayed on login pages." Okay. So Marc, I think what you're missing here is that the reason favicon.ico is a good test is that it is almost certainly known to be an image on the root of any server. So without needing to customize a query per site, where you would just go get some image on the site, and you could certainly do that, you could know that Google has this image on their logo or something, and you could query that image. But most sites are going to have a favicon.ico image.

The point is, images are allowed to be pulled by scripts across domain, meaning that you can go get some other site's .ico image. And all we're wanting from that is to obtain the cookie for that site and that user. And cookies are always offered with any asset from a site. So it doesn't matter if it's anonymous or not. All it has to be is any image. And so favicon.ico is a good one.

And this was sent to me and to you, Leo. Kind of a cool project that a Luis Zepeda performed for us. He said: "Hey, Leo. I remember you asked @SGgrc if he believed that security was getting better across all these years of Security Now!. Well, I ran some natural language processing and sentiment analysis tools" - and here there's a graph, you should put it up on the screen - "sentiment analysis tools across all the transcripts, and his analysis shows a slight downward trend. It's getting slightly worse."

Leo: Downward is bad.

Steve: Yeah. So from Episode 0 out through, looks like he went all the way through, like Episode 636.

Leo: Look at this outlier up here. We were in a really good mood that day.

Steve: That must have been a happy day, yes. Woohoo.

Leo: So these are, like, negative words and thoughts.

Steve: Yes, correct.

Leo: Wow.

Steve: And so basically some things are positive, and some things are negative. And maybe it actually reflects my amount of caffeination. I'm not sure.

Leo: Well, that's pretty - it's a pretty random distribution.

Steve: It is. It is, very. It's only a very slightly downward trend.

Leo: Very slight, yeah. Wow.

Steve: Very cool. But thank you, Luis.

Leo: That's brilliant. I know we have some geeky listeners. Wow. That's amazing.

Steve: And Ronald A. Suchland said: "I cannot find anything to stop Leak Test." What?

Leo: Huh?

Steve: "Win7 64-bit. Lavasoft locks up the computer." Okay. So Leak Test is something I wrote many moons ago to demonstrate the concept of outbound blocking. That is, the idea - and this was because I liked the idea that something in your computer could be caught leaking behind your back. And in fact it was because I was an early tester of ZoneAlarm that ZoneAlarm warned me of the Aureate adware which was connecting without my knowledge or permission behind my back. And it was like, what? What? What is this DLL that I didn't give permission to doing talking to somebody I didn't say it could talk to?

So I created Leak Test as a simple app that anyone could use to see whether their system would detect outbound connections. So it's really sort of obsolete. The by-default things in your computer are allowed to create web connections to the outside world. So, yeah. If you had a firewall which was in the machine, where you had set it to "Deny all unless I explicitly accept," then Leak Test would fail to make its connection, and maybe you would get warned. But that would be very burdensome. We did that once upon a time with ZoneAlarm. And now we've sort of loosened things up.

The firewall that is in Win7 64-bit will do that. If you told it to, you could lock it down tight and tell it to deny every connection from anything you didn't ask, explicitly permit. But you'd pull your hair out in no time. I mean, if you've ever done a netstat on your computer, netstat -an, oh, my goodness, it shows what's going on. And we've just lost control of all this now. So it's like trying to run NoScript these days on a browser. You just can't. Everything is scripting, and everything is phoning home. And we just have to trust that nothing evil is being done behind our back.

And finally, let's see. Oh, this is Chris in Germany, a "Comment on Steve's comment last episode." He said: "Hello, Leo and Steve. Steve, you commented in the last episode that users should never download anything, even from trusted sources."

Leo: I don't think we said that.

Steve: No, I didn't. But I want to clarify. "Does that include the show notes you and Leo upload to your show?"

Leo: No, never download anything.

Steve: "To be fair, I don't think this advice is feasible for the average user and sounds Stallman-esque. Could you elaborate on that further? How am I supposed to use the Internet without downloading anything?" And he sounds German, doesn't he. "Thanks a lot. I appreciate the podcast and your experience."

Chris, let me explain. What I said was do not ever download anything offered to you that you didn't go looking for. That's the distinction, and I would argue probably the single most valuable piece of advice available today. That's what sites do when they say, "Oh, you need to download this new font because you want to see this page properly." Or, "Oh, we want to show you this amazing person making pizza, but you need to update your version of Adobe Flash. Click here." Those are things being pushed on you, pushed to you. And to that you want to say no. It is almost guaranteed to be malware.

So if you believe you need a new version of Adobe Flash, go to Adobe and get it, after thinking five times about whether or not even that is a good idea. The point is never accept candy from a stranger, or a download link from some random site, even one you trust. Yes, download our show notes. Download SQRL as soon as it's available from GRC. But go get it. Don't go to a site that says, "Oh, you need SQRL. Click here, please." Come to GRC and ask for it. It's free. But never download something that some site is offering to you. That's the point I'm trying to make because that's a form of social engineering. Many sites try to take advantage of the naivet of users who are, like, "Oh, I didn't realize that was obsolete. Okay, I'll click here to update." I mean, the problem is it's going to be successful most of the time.

Okay. So last week our often-quoted guru and security expert and crypto person whose books I have behind me on my shelf, Bruce Schneier, was asked to testify before the House Energy and Commerce Committee on his feelings about the Equifax hack. There is on his site - his current blog is www.schneier.com, S-C-H-N-E-I-E-R dot com - there's a link to the video of his testimony, if you want to watch it. But I want to share some of the points he made because they were good.

And he starts out a little bit with his CV: "Mr. Chairman and Members of the Committee, thank you for the opportunity to testify today concerning the security of credit data. My name is Bruce Schneier, and I am a security technologist. For over 30 years I have studied the technologies of security and privacy. I have authored 13 books on these subjects, including 'Data and Goliath: The Hidden Battles to Collect Your Data and Control Your World,' published in 2015 by Norton."

He says: "My popular newsletter Crypto-Gram and my blog Schneier on Security are read by over a quarter million people. Additionally," he says, "I am a Fellow and Lecturer at the Harvard Kennedy School of Government, where I teach Internet security policy, and a Fellow at the Berkman Klein Center for Internet and Society at Harvard Law. I am a board member of the Electronic Frontier Foundation, Access Now, and the Tor Project; and an advisory board member of Electronic Privacy Information Center and VerifiedVoting.org. I am also a special advisor to IBM Security and the Chief Technology Officer of IBM Resilient."

So obviously he's got a beautiful CV that demonstrates to these guys who have no clue

which end is up that this is a guy whose opinion is informed. He says: "I am here representing none of those organizations and speak only for myself based on my own expertise and experience. I have eleven main points." And I'm going to skip them toward the end, but I want to share the main salient ones.

He says: "One, the Equifax breach was a serious security breach that puts millions of Americans at risk." We all know that, but he wants to establish some ground. "Equifax reported that 145.5 million U.S. customers, about 44% of the population, were impacted by the breach. That's the original 143 million plus the additional 2.5 million disclosed a month later. The attackers got access to full names, Social Security numbers, birth dates, addresses, and driver's license numbers.

"This is exactly the sort of information," Bruce says, "criminals can use to impersonate victims to banks, credit card companies, insurance companies, cell phone companies, and other businesses vulnerable to fraud. As a result, all 143 million US victims are at greater risk of identity theft, and will remain at risk for years to come. And those who suffer identify theft will have problems for months, if not years, as they work to clean up their name and credit rating.

"Two, Equifax was solely at fault." He says: "This was not a sophisticated attack. The security breach was a result of a vulnerability in the software for their websites, a program called Apache Struts. The particular vulnerability was fixed by Apache in a security patch that was made available on March 6, 2017 and was not a minor vulnerability. The computer press at the time called it 'critical.' Within days it was being used by attackers to break into servers. Equifax was notified by Apache, US-CERT, and the Department of Homeland Security about the vulnerability and was provided instructions to make the fix.

"Two months later, Equifax had still failed to patch its systems. It eventually got around to it on July 29. The attackers used the vulnerability to access the company's databases and steal consumer information on May 13, over two months after Equifax should have patched the vulnerability. The company's incident response after the breach was similarly damaging. It waited nearly six weeks before informing victims that their personal information had been stolen, and that they were at increased risk of identity theft. Equifax opened a website to help aid customers, but the poor security around that - the site was a domain separate from the Equifax domain - invited fraudulent imitators and even more damage to victims. At one point, the official Equifax communications even directed people to that fraudulent site."

He says, finishing point two: "This is not the first time Equifax failed to take computer security seriously. It confessed to another data leak in January 2017. In May 2016, one of its websites was hacked, resulting in 430,000 people having their personal information stolen. Also in 2016, a security researcher found and reported a basic security vulnerability in its main website. And in 2014, the company reported yet another security breach of consumer information. There are more.

"Three," he says, "there are thousands of data brokers with similarly intimate information, similarly at risk. Equifax," he says, "is more than a credit reporting agency. It's a data broker. It collects information about all of us, analyzes it all, and then sells those insights. It might be one of the biggest, but there are 2,500 to 4,000 other data brokers that are collecting, storing, and selling information about us, almost all of them companies you've never heard of and have no business relationship with.

"The breadth and depth of the information the data brokers have is astonishing. Data brokers collect and store billions of data elements covering nearly every U.S. consumer.

Just one of the data brokers studied holds information on more than 1.4 billion consumer transactions and 700 billion data elements, and another adds more than 3 billion new data points to its database each month. These brokers collect demographic information: names, addresses, telephone numbers, email addresses, gender, age, marital status, presence and ages of children in household, education level, profession, income level, political affiliation, cars driven, and information about homes and other property. They collect lists of things we've purchased, when we purchased them, and how we paid for them. They keep track of deaths, divorces, and diseases in our families. They collect everything about what we do on the Internet."

He says: "Number four, these data brokers deliberately hide their actions and make it difficult for consumers to learn about or control their data." He writes: "If there were a dozen people who stood behind us and took notes of everything we purchased, everything we read, searched for, or said, we would be alarmed at the privacy invasion. But because these companies operate in secret, inside our browsers and financial transactions, we don't see them, and we don't know they're there.

"Regarding Equifax, few consumers have any idea what the company knows about them, who they sell personal data to, or why. If anyone knows about them at all, it's about their business as a credit bureau, not their business as a data broker. Their website lists 57 different offerings for business - products for industries like automotive, education, healthcare, insurance, and restaurants. In general, options to 'opt-out' don't work with data brokers. It's a confusing process and doesn't result in your data being deleted. Data brokers will still collect data about consumers who opt out. We will still be in those companies' databases and will still be vulnerable. It just won't be included individually when they sell data to their customers.

"Five," he says. "The existing regulatory structure is inadequate. Right now there is no way for consumers to protect themselves. Their data has been harvested and analyzed by these companies without their knowledge or consent. They cannot improve the security of their personal data and have no control over how vulnerable it is. They only learn about data breaches when the companies announce them, which can be months after the breaches occur, and at that point the onus is on them to obtain credit monitoring services or credit freezes. And even those only protect consumers from some of the harm, and only those suffered after Equifax admitted to the breach.

"Right now, the press is reporting dozens of lawsuits against Equifax from shareholders, consumers, and banks. Massachusetts has sued Equifax for violating state consumer protection and privacy laws. Other states may follow suit. If any of these plaintiffs win in the court, it will be a rare victory for victims of privacy breaches against the companies that have our personal information. Current law is too narrowly focused on people who have suffered financial losses directly traceable to a specific breach. Proving this is difficult. If you are the victim of identity theft in the next month, is it because of Equifax, or does the blame belong to another of the thousands of companies who have our personal data? As long as one can't prove it one way or the other, data brokers remain blameless and liability free.

"Additionally, much of this market in our consumer data falls outside the protections of the Fair Credit Reporting Act. And in order for the FTC (Federal Trade Commission) to levy a fine against Equifax, it needs to have a consent order and then a subsequent violation. Any fines will be limited to credit information, which is a small portion of the enormous amount of information these companies know about us. In reality, this is not an effective enforcement regime. Although the FTC is investigating Equifax, it's unclear if it has a viable case."

And so anyway, I won't go on. "Number six," he says, "the market cannot fix this because we are not the customers of the data brokers." As we know, we are the products which the data brokers sell. So this has perverse incentives. The data brokers are selling to companies that want the information. So this doesn't, in this system, traditional market forces don't work to apply pressure. The customers want the information, want it to be easy to get, don't want us to be able to block it from their access. So as a consequence, it has been made hard for us to do this. And in fact he makes the point that financial markets reward bad security.

He writes: "Given the choice between increasing their cybersecurity budget by 5% or saving that money and taking the chance, a rational CEO chooses to save the money. Wall Street rewards those whose balance sheets look good, not those who are secure. And if senior management gets unlucky and a public breach happens, they end up okay. Equifax's CEO did not get his $5.2 million severance pay, but he did keep his $18.4 million pension. Any company that spends more on security than absolutely necessary is immediately penalized by shareholders when its profits decrease."

And he finishes: "Even the negative PR that Equifax is currently suffering will fade. Unless we expect data brokers to put public interest ahead of profits, the security of this industry will never improve without government regulation." Anyway, so "Number seven, we need effective regulation of data brokers. Number eight, resist the complaints from the industry that this is too hard." He notes that credit bureaus and data brokers and their lobbyists and their trade association representatives will claim that these measures are too hard.

He says: "They are not telling you the truth." He says: "Take one example, credit freezes. This is an effective security measure that protects consumers. But the process of getting one and of temporarily unfreezing credit is made deliberately onerous by the credit bureaus. Why isn't there a smartphone app that alerts me when someone wants to access my credit rating and lets me freeze and unfreeze my credit at the touch of the screen? Too hard? Hardly. Today you can have an app on your phone that does something similar if you try to log into a computer network, or if someone tries to use your credit card at a physical location different from where you are."

He says: "Moreover, any credit bureau or data broker operating in Europe is already obligated to follow the much more rigorous EU privacy laws. The EU General Data Protection Regulation will come into force, requiring even more security and privacy controls for companies collecting and storing the personal data of EU citizens. Those companies have already demonstrated that they can comply with those more stringent regulations."

Anyway, so really, really good testimony from Bruce. He finishes with number 11, saying: "We need to do something about it. Yes, this breach is a huge black eye and a temporary stock dip for Equifax - this month. Soon, another company will have suffered a massive data breach, and few will remember Equifax's problem. Does anyone remember last year when Yahoo admitted that it exposed personal information of a billion users in 2013 and another half billion in 2014?" He says: "Unless Congress acts to protect consumer information in the digital age, these breaches will continue."

Finally: "Thank you for the opportunity to testify today. I will be pleased to answer your questions." And Bruce then did that. So bravo for having someone who understands the problem, who understands security, and who understands that we could easily, if they chose to, give us the technology, at least in these cases, to manage the availability of our credit far more usefully than we have today.

Leo: And yet nothing will ever happen.

Steve: Nope.

Leo: I'm impressed that Congress, or staffers probably, had the good sense to invite Bruce, though. That's good.

Steve: Yup, yup. It certainly won't happen without it. And so maybe something will come of it. We can hope.

Leo: I love the idea. And of course it would be a simple thing to have an app that says, "Hey, you want to unfreeze your credit? Somebody wants a credit report." Say yes, boom, done.

Steve: Yeah, well, it would be like so-and-so who you're renting an apartment from wants to verify your credit.

Leo: Or Google or Microsoft want - you just tried to log in. You approve it? Yes.

Steve: Yup, exactly.

Leo: It's an easy thing to do, but of course it impedes their ability to profit on selling your credit report.

Steve: Creates a little bit of friction, yes.

Leo: And they don't, well, it stops it cold. If you have a credit freeze on your account, they can't make any money on you.

Steve: Right.

Leo: Ah, well. Such is life. Steve has done it again, my friends. In a mere two hours and a few odd minutes, he has totally - well, less than that, actually. About an hour 50. He's totally changed our perception of the world around us. And now we put him back in his box for another week, and he will continue to gather information for next week's episode.

Steve: I'll do it.

Leo: You can find this and every episode, how many have we done here, 637, at GRC.com. He's got transcripts there, too. That's where that sentiment analysis came from. Those are freely available, downloadable, written not by a machine, but an actual human person named Elaine. She does a great job, so take advantage of those. He also has 64Kb audio files there. And when you're there, make a little yabba dabba doo happen in Steve's heart by buying a copy of SpinRite, the world's finest disk hard drive recovery and maintenance utility. That's his bread and butter.

Steve: It is that.

Leo: Without that, he would have no coffee. So please...

Steve: Keep me caffeinated.

Leo: Keep him caffeinated. God only knows what would happen if he weren't. At this point I think the caffeine's the only thing holding his veins open. You've got to help it out. Help a kid out. You could buy a copy of SpinRite, or you could turn the page. Don't turn the page. You could also find audio and video. Oh, you really don't want to see the video of this. You can get video of this fine show, really riveting video. The only thing I can say about the video that's good about it is, because very little is moving in the shot, basically Steve's lips and those blinky lights behind him, the compression is excellent.

Steve: Fabulous-looking, yeah.

Leo: Fabulous and highly compressed video available at TWiT.tv/sn. You can also subscribe and watch at your heart's content on your favorite device. TWiT.tv/sn. And if you want to watch live, do it every, well, best to come by around 1:30. We usually get to Steve a little bit later, but around 1:30 p.m. Pacific on Tuesday afternoons. There's another one. Did you hear that?

Steve: That was a yabba dabba doo. Somebody's...

Leo: Somebody yabba dabba doo'd you.

Steve: Thank you very much, listener. Thank you.

Leo: That's somebody listening live right now and said, "I'm going to do that. That's a good idea."

Steve: Thanks.

Leo: Let's make more yabba dabba doos happen. Sometime between 1:30 and, oh, 4:10 Pacific, that's 4:30 Eastern time. That's 21:30 UTC. You can come by, watch the show, be in the chatroom at irc.twit.tv. Yeah, if you shut off the blinky lights, the file size will go down another 20%. Bill's pointing that out. I think he's got a good point. That's really - that's killing the key frame there. Killing it. We will see you next time, Mr. Steve Gibson, on Security Now!.

Steve: Thank you, my friend. Next week.