STARTTLS Everywhere

Description: This week we discuss another worrisome trend in malware, another fitness tracking mapping incident and mistake, something to warn our friends and family to ignore, the value of periodically auditing previously granted web app permissions, and when malware gets picky about the machines it infects. Another kind of well-meaning Coinhive service gets abused. What are the implications of D-Link losing control of its code-signing cert? There's some good news about Android apps. iOS v11.4.1 introduces "USB Restricted Mode," but is it? We've got a public service reminder about the need to wipe old thumb drives and memory cards. What about those free USB fans that were handed out at the recent North Korea/U.S. summit? Then we take a look at email's STARTTLS system and the EFF's latest initiative to increase its usefulness and security.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-671.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-671-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Coming up, we're going to talk about a hack to Fortnite cheaters. Couldn't happen to a nicer bunch. Also why your browser might be freezing. More privacy leakage from a fitness app. And at the end Steve's going to talk about a new way, a new proposal to keep email secure. Actually, it's an old proposal that might be getting new life. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 671, recorded Tuesday, July 10th, 2018: STARTTLS Everywhere.

It's time for Security Now!, the show where we cover your security and privacy, explain it all in-depth, thanks to this guy right here, Steve Gibson of the GRC Corporation.

Steve Gibson: I love it how you turn around to look at me, Leo. It's like...

Leo: Well, you're right behind me. You're over my shoulder.

Steve: I'm right there. I'm right there.

Leo: If I don't look around, I don't see your salute.

Steve: That's right. Got to have the Vulcan whatever it is, salute.

Leo: The ancient Yiddish salute, yeah.

Steve: So as we expected, when I added an extra "T" to STARTTTLS, and it's actually just STARTTLS, for those who were...

Leo: You could say it's STAR TTLS.

Steve: Or I could be stuttering, STARTTTTTTLS.

Leo: There's a lot of T's in there.

Steve: I can't count them all. As expected, we have a lot of interesting fun stuff, but nothing dramatic that pushed back my intention, as I mentioned last week, to talk about the latest EFF initiative, STARTTLS Everywhere, which should remind us of HTTPS Everywhere. And there are a lot of overlaps and similarities, but of course some differences. This is the effort to push email encryption further towards, well, let's just say usefulness because it isn't very right now. And this is not PGP or S/MIME or you encrypt it, then you send it. This is an attempt to shore up sort of this weird intermediate solution which had actually kind of already been abandoned. But as with everything on the Internet, if it kind of works, then it's really hard to kill. So rather than trying to kill it, they're trying to fix it.

So we're going to end the podcast talking about that, sort of understand where we are with not additional encryption, which of course an end user can always decide they want to employ, which would give you true end-to-end protection. As a consequence of the way email works, this doesn't, even in the best of times, you don't get end-to-end encryption, I mean, even using the latest solutions and systems, because email is a multipoint hop system, store and forward, by design. And so essentially any servers along the way are man in the middles who get your unencrypted email. Again, when everything is working perfectly. So this is why I've never been super excited about the prospects of encrypting email unless an individual goes to extremes. But anyway, we're going to wrap up with that.

First, we're going to talk about another new and emerging worrisome trend in malware, yet another fitness tracking mapping incident and mistake, something to warn our friends and family to ignore if it happens to them, the value of periodically auditing previously granted web app permissions, and some malware that has actually become picky about which machines it infects and what it does depending upon what it finds. We have another kind of well-meaning Coinhive service which didn't take long to get abused.

We're going to look at the implications of D-Link having lost control of its code-signing cert, which did happen. Some good news about Android apps. We will talk, as you did on the previous podcast, Leo, about last night's iOS update to v11.4.1, which among other things introduced something new that the user has control over, known as a "USB restricted mode," and whether it actually is, or at least something to be aware of as a wrinkle in the way it works. We have a public service reminder about the need to wipe old thumb drives and memory cards, I mean, wipe them proactively before you let them out of your control, and what a team in the U.K. discovered when they just bought a bunch.

Also, the Picture of the Week is just - I got a kick out of this, and our listeners will. It turns out that free USB-powered fans were handed out at the recent North Korea-U.S. Summit, where our fearless leader met theirs. And of course it really upset the security mavens, who were like, wait a minute, folks, those were made in China. Please don't plug them into your phones without - it's like, what could possibly go wrong?

Leo: Was this about the one that just happened, or the one with Trump earlier?

Steve: The one - it was the 14th. It was the 14th.

Leo: Yeah, Pompeo was there talking to them last week.

Steve: Right, right.

Leo: So this is...

Steve: So this was the first one.

Leo: The first one.

Steve: And then we...

Leo: I could just see some - I could see, you know, god, you can just see Donald Jr. picking this up, saying, oh, this is great, this is great. That'd just keep me cool in the summer.

Steve: Well, and it looks like it was nicely designed from the photo, which is our Photo of the Week. I used the caption "Free Chinese-made USB fans handed out during the recent North Korea/U.S. political summit. What could possibly go wrong?"

Leo: So I'm guessing on the other side of that wire is a USB plug. You plug it into your laptop, it's got a little gooseneck so that you can aim the fan at you while you're sitting in front of your laptop.

Steve: Well, actually to me it looks like the fan module itself is a USB Type C.

Leo: Get it, I get it, yeah, yeah. There's a Lightning on it, as well, yeah.

Steve: And then they have a little - exactly. So you could plug...

Leo: So it's for your phone.

Steve: Exactly. And you would plug it, and the way it's tilted, you would plug it into the bottom of your phone where the power connector is, and that would give you a fan at the bottom of your phone facing your face. So it's, like, convenient. And you can imagine the security people were quite upset.

Leo: USB-protected mode on this one.

Steve: You need a USB condom is what you need.

Leo: Well, do you, though? I mean, I guess phones have data access. But there's not, I mean, what would they do with this? Could it infect a phone?

Steve: Yeah, it could have malware.

Leo: Yeah.

Steve: And that was the concern. Anyway, we will talk about all of that in the fullness of time, Leo.

Leo: Yes, yes.

Steve: Before we let our listeners go off about their normally scheduled lives. They are captive until then.

Leo: We like to keep them captive. On we go, Mr. Gibson.

Steve: So I've heard you talk about Fortnite a lot.

Leo: Yes, love it.

Steve: Amazed by the amount of money that those guys make for a game that they give away for free. Predictably, there would be add-ons, and we've seen this with all games which achieve stardom and fame, which are cheats of various sorts.

Leo: Yeah. It's a big problem with multiplayer online games because, if you're playing against somebody who cheats against you, that really sucks.

Steve: Yeah, yeah. And it is, you know, it's technology which is prone to manipulation.

Leo: Right.

Steve: So there's a service known as Rainway which is - it's sort of similar to PlayStation now. It's recently out of beta, earlier this year. It's a videogame streaming service that allows users to run games on their PC and play them on devices over an Internet connection, play them on other devices over an Internet connection. So sort of like Remote Desktop for gaming.

Okay. Last week Rainway engineers, some of whose customers were naturally playing Fortnite because it's so popular, began detecting infections in their users' systems when the Rainway server logs began reporting hundreds of thousands of errors. I think 308 or something thousand errors that they had logged in a short period of time. The errors were the result of ads that had somehow been injected into user traffic. Rainway uses whitelist filtering to permit its customers to connect only to approved URLs, so those connection attempts were being caught and blocked.

And note that, like, for non-Rainway users, this wouldn't have happened. So it was just sort of a coincidence of the fact that some Fortnite users had downloaded a cheat whose behavior was being, like, raising flags in the streaming service that they happened to be using for playing videogames. And that service had the foresight to whitelist specific URLs, blocking other ones, which then raised the flag. Otherwise this would have - who knows how long it would have taken to catch this.

So it turns out that the attempts that were caught were fraudulent content injected into the sessions of these Rainway customers, attempting to connect to two different ad platforms, Adtelligent.com and SpringServe.com. And there was also unauthorized JavaScript that was bound to those ads. Which, again, this is sort of like, what? The Rainway engineers were like, okay, what is going on here? So this strongly suggested to Rainway that something was going on.

Rainway examined the profiles of the affected users, looking for common threads, because initially all they were seeing was just errors. They didn't know what their users were doing that was causing this trouble. They had no hardware in common. Their ISPs were different. All of their systems were up to date. But one thing stood out. All of them were playing Fortnite.

So then, suspecting that some malware might have been spread by one of the many Fortnite cheating hacks available, the Rainway guys started downloading thousands of add-ons, looking for something that might be causing this trouble. And they found it. They found an add-on which was promising in-game currency, I guess it was called V-Bucks is the currency within that world, and also promising to equip users with an "aimbot" which would automatically aim the player's gun at their opponents without any need for particular precision by the player. Oh, and what they did was they scanned all of these different add-ons, looking for the strings which they were seeing being emitted and captured in their logs. That's how they found this one.

So they then ran the add-on in a virtual machine and discovered that, among other things, it installed a self-signed root certificate into the user's playing platform, into their PC. And of course, as we know, that would enable a man-in-the-middle attack on every HTTPS website the user visited. And as they continued to look at it more closely, they discovered that it was indeed intercepting - it was using this newly installed self-signed root cert to perform a man-in-the-middle attack on all of the web pages that the game players were visiting on the fly to insert references to ad platforms.

So they reported the rogue malware to the service provider who was hosting it, and that provider immediately removed the malware and let them know that it had been downloaded 78,000 times. So there are 78,000 instances of Fortnite players who had downloaded a cheat and got themselves hacked as a process, and had a rogue self-signed certificate stuck into their root certificate store in order to perform this basically

ad fraud with their system. And at the time of their reporting, Rainway reported 381,000 interceptions of blocked URL attempts in their own logs. And I'm sure only a fraction, tiny fraction of Fortnite's users happen to also be using the Rainway service. So the infection is probably massive.

The Rainway guys also reported their discovery to the Adtelligent and SpringServe ad platforms, who identified and pulled the ads from their platform. And in Ars Technica's coverage of this, Dan Goodin reached out to Epic Games, who is the maker of Fortnite, but Epic declined to comment.

So I'm beginning to see this behavior, and I wanted to sort of put it on our listeners' radar. When I say "this behavior," what I mean is instances of self-signed root certificates being stuck in people's root stores. So I wanted to make sure that everybody understood what that meant. A CA certificate that we often talk about how there are now thousands of them that we trust, what those certificates are, are self-signed certificates, that is, it is a certificate that no higher authority vouches for. The authority is essentially vouching for itself. So it signs its own certificate so that the certificate has a valid signature.

But the signature is valid only because the certificate is in the root store. So when Windows or whatever platform, Linux, iOS, anything that is using root certificates, when the system checks the certificate, it finds it in its own root store because that's where the certificate is. So it's automatically trusted. But more importantly, the certificate is also the root of a chain of trust, meaning that any other certificates that come along are checked against the certificates in the root store to see whether they should be trusted.

So when malware drops a self-signed certificate into the user's root certificate store, what's happening is like really bad. It's what you don't want because, as we've talked about, we already sort of have a fragile public key infrastructure based on trusting anything that is signed by any of the certificates in our root store. And so this is where the CAs that have gotten in trouble, like famously Symantec did a couple years ago, and essentially just lost their certificate signing business as a consequence, if their certificates can't be trusted, they get kicked out of the root store, and it's game over for them. So as I said, I'm running across more instances of this, which is why my radar perked up when I saw that this is now the behavior we're seeing.

Once upon a time, before all web traffic was encrypted, that is, when most web traffic was still HTTP under the somewhat mistaken belief that adding security was too processor intensive, it would slow things down, that really wasn't, except in the very early days, that wasn't the case. It was just inertia. It was just everything seems to be running fine. Again, nothing seems wrong - except when we finally woke up to the fact that our session tokens were in plaintext cookies that allowed a passive eavesdropper to jump onto and impersonate a user simply by grabbing their cookie that was passing in the clear in any open WiFi scenario, for example, and be logged on as them. So as we know, pretty much now, thanks to another EFF initiative, Let's Encrypt, but also just even before then, just a raising of the awareness of the importance of encryption, suddenly it was no longer feasible to alter web pages and inject content.

So unfortunately, what this has done is it's put pressure on the system which is trying to protect us, with the consequence being that malware is now routinely - or maybe not yet routinely, but again I think we're going to see this as an emerging trend - is saying, okay, fine. All of the software for creating a little HTTPS intercepting server is open source and in the public domain now. All we need to do in order to terminate the connection ourselves, just like a corporate middlebox does, which we've often talked about, is drop a certificate into the user's machine, just like a corporate middlebox does that wants to filter for antivirus and provide content filtering for the people being protected within the network.

So we've talked in the past about auditing our certificate root stores. And in fact I played with one for a while that dynamically looked to see what certificates were being used and raised a flag. I ended up having to abandon that particular tool because Google is themselves a CA, and I'm a user of Google properties, and they were constantly churning their own certificates. They have every right to do so, but it was just causing - it was raising false positives all the time of certificates changing. And so I finally just said, okay, well, this isn't working as it is.

But I think that what we're going to need to see, we're going to need to have our certificate stores hardened more than they are because they shouldn't need to be changed often. I mean, I would argue that there's really not a good reason for the certificate store to be populated and managed by anything other than the official maintenance system of our OS vendor. So, for example, in the case of Windows, it should only be Windows Update that has the ability to mess with that. I think only due to, again, historical reasons is it something that is browsable, and you can poke in and look at and decide if you want to mess with it.

I was doing that myself just a couple weeks ago, needing a localhost server on my system in order to do some work. And it wasn't hard to just drop a self-signed localhost cert into my machine in order for a server using it to be trusted. So there are valid reasons. But you should have to jump through all kinds of hoops to do that. That should not be something that random malware that gets onto your system is able to do because I think this is, as I said, this is the next thing we're going to see is more instances of this happening in the future. I think this is something we need to lock down, and so far that hasn't happened.

In another piece of global fitness tracking mapping gone wrong, our listeners will recall that I think it was in 2016 we talked about the Strava fitness tracking app which was just sort of publishing all of the positioning information of its users because everyone kind of thought that would be cool, to look at maps of where people were using their app. Unfortunately, it turned out that there were people on secret military bases going out for a jog around camp who were being logged, and satellite-looking maps had this Strava fitness tracking data overlaid on it, and suddenly there was a sort of suspicious lot of activity in areas of the desert where nothing was shown on the maps. And so anyway, we talked about it at the time.

Well, it turns out that some investigations carried out by reporters from a Dutch newspaper, De Correspondent, and online investigations group Bellingcat have found another instance which caused Polar, the well-known - I think they're Dutch, aren't they?

Leo: I don't know.

Steve: No. I don't remember. Polar is Finnish. Finnish, yes.

Leo: Finnish, yeah, that's right,

Steve: Yeah. So what they found has caused Polar to suspend its global activity mapping feature. It's kind of not Polar's fault. But, well, anyway, so here's what happened. The two groups of reporters discovered that Polar Flow, which is one of Polar's apps, was allowing anyone access to a feature called Explore, which is Polar's activity map. The data exposed on this map included a user's past activity - running, biking routes and so forth - but included the user's personal details, such as their heart rate, various fitness attributes, and more.

Well, what Polar did was they made the mistake of exposing the username and personal details of each individual for each individual activity and route. And upon digging into it deeper, it looks like the user needs to explicitly allow that information to be shared, but they probably didn't understand fully the consequences. The reporters tracked down, were able to thanks to this disclosure, the real world identities of specific intelligence and military personnel. The journalists used this feature to search the map for the location of known military bases and intelligence agencies' headquarters and training grounds. They were then able to identify the people who reported fitness activity history at those locations.

And in several cases the researchers were able to identify the usernames, which led back to real-world identities, either because the military and intelligence agents used their real names for the Polar app or because they reused usernames that were associated with them elsewhere, and the reporters were able to make the connections. Oh, and also in some cases the same routes which they ran while jogging on base also contained the jogging and biking routes in other locations for the same person, which then allowed that linkage exposing what looked like their user's home address.

So again, in response to these revelations, Polar immediately shut down the global activity mapping feature, but they then did seek to clarify that this Polar Flow app does not expose the user's activity and username by default. Polar said that these details are shared only based on an opt-in system, and the data exposed via its activity map had to be willingly shared only by - or had to be and was being shared only by some of its users, with the majority of their users' data remaining private.

So I guess I would say that this is clearly a privacy concern, and anybody who is using this really needs to have it be made clear. I mean, there's some tension here; right? Polar, much like Strava, wants to brag about how many people are using their stuff. It's people all over the world. Look at how busy everybody is getting fit and healthy. The flipside is, I mean, it is an information disclosure. And so people, I think, who are saying, yes, I want others to be able to see my little red lines where I'm running, and/or my username and other data, I want to share that, really need to be made aware of the downside consequences from a privacy standpoint.

Leo: I don't think you can blame Polar Flow for that. I mean, yeah, maybe it's good for Polar to show how many people use their app. But that's the point of it. And Apple does the same thing with their activity rings. You can share with other people, and then you compete with them and so forth.

Steve: Do you share it with specific people in the Apple case, though?

Leo: Yeah. In the Apple case...

Steve: Because this was shared globally.

Leo: Yeah, but...

Steve: This was a global and, you know, share...

Leo: So there were two things that were wrong. I mean, clearly, if you're an NSA intelligence officer, and you're publicly sharing your run route, you need to kind of go back to school and learn some op sec stuff.

Steve: That's right.

Leo: But there seems to have also been a problem...

Steve: You were absent for that day of training when they said...

Leo: Uh, what? But, I mean, it's actually not a surprise. There are a lot of military folks in great shape. And so they're exercising. So, you know, clearly they're at fault for making it public. The problem really also was there was an API that was exposing this, and they used sequential index numbers for customers. So you could just query the API and get this information. And that's clearly Polar's fault.

Steve: So a poor design from an anonymizing standpoint.

Leo: Yeah, yeah. But, I mean, there's lots of apps that, you know, Runkeeper does this, too, where you share your run. And people like to do that. They like to, you know.

Steve: Yeah, yeah.

Leo: They probably shouldn't be doing it if they're in a secret location.

Steve: Probably not if your location should not be disclosed.

Leo: Yeah, especially since this whole thing came up already, and you'd think...

Steve: Yes, exactly, two years ago it made the news.

Leo: ...the memo would have gone out.

Steve: And it couldn't have been any more specific.

Leo: It was a running program. It was exactly the same.

Steve: Right. Okay. So Leo, on your Tech Guy show for Saturday and Sunday for three hours on the weekends, there's no question that people have called you asking what this is. And it uses an interesting hack which I wanted to talk about and just sort of put it on our listeners' radar. I doubt that any of our listeners would get caught out by this. But

probably they have friends and family who might. So it turns out that freezing someone's browser is a thing.

Leo: Oh, that's interesting.

Steve: Yeah.

Leo: Because we do get calls about that, yeah.

Steve: Yes. It is a deliberate tactic on the part of bad guys to sell an assertion they're making which users might otherwise just kind of blow off. It's that "Your computer has been infected with malware. We're Microsoft. And so we have locked your computer. Please call this toll-free number" - and, you know, it's 1-888 something or other - "in order to talk to one of our technical support people so that we can help you disinfect your machine and get you going again." And so if it just came up on a web page, given all the nonsense people encounter on the 'Net, people would be inclined to go, eh, what, and just close the page.

But what is being done is the browser is locked completely. The UI is frozen, which plays into the warning that is being presented and helps to sell it to people who would otherwise think, eh, this seems like a scam. So, okay. Back in February Malwarebytes first described a scam using a particular cross-browser web API, which is window.navigator.msSaveOrOpenBlob, which is cross-browser supported. It's the means for a browser to save, as it sounds, a blob of just sort of opaque data for its own purpose on the user's machine. And it provides locally storing and retrieving these blog files.

But it's an asynchronous operation. That is to say, the call is made to save or open the blob, well, in the case of save. And the JavaScript goes on, assuming that that's going to be done. So this allows JavaScript to do whatever it wants to do while this save is occurring in the background. Well, that also allows this call to be placed into a tight loop, with the blob save requests quickly becoming backlogged in the browser with the OS unable to do the work of creating lots of little files quickly enough. This pins the system CPU at 100%, and within a few seconds completely shuts down the browser's UI. You can't close the tab. You can't even click the close button on the browser. You can't get out. The whole browser is locked. And not only does the browser become nonresponsive, but in some cases the system does, too.

So of course people get freaked out by this. So Chrome v65 in February fixed the problem, and then it came back in Chrome 67 and is today once again being actively exploited. And it's not just Chrome. The Microsoft browsers, both Edge and IE, are not affected by this. I'm just guessing, I haven't looked, but they probably just don't bother to support this web API. They generally are lagging a little bit behind. Or maybe they just do it correctly. But Firefox and other non-Microsoft browsers, including Opera, Vivaldi, and Brave, are all caught out by this. And I didn't see any mention of Safari one way or the other. So I'm not sure over on the iOS platform.

Leo: Well, I'm thinking, because it says "ms blob," it might be Microsoft specific.

Steve: Except that it's Chrome, Firefox, Opera, Vivaldi, and Brave. So I'm not sure...

Leo: Yeah. [Crosstalk] have to be on Windows, I'm saying.

Steve: Ah, okay. That's a possibility, yeah.

Leo: Because I don't know what "ms blob" would do on a Macintosh.

Steve: Yeah. Anyway, so this behavior has returned, and lots of people are reporting that their browsers are freezing with this notice. So first of all, I'm sure our listeners know that in Windows Ctrl-Alt-Del will bring up a menu. Among the items there is Task Manager. And you can then use Task Manager to terminate the visible processes under your browser name. You'll see a bunch of Firefox.exes, a bunch of Chrome.exes, or whatever. And under macOS there's a Force Quit that macOS gives you in order to just kill the browser, in order to get this thing to stop. So I just wanted to sort of put it on our listeners' radar. Again, I'm sure that our listeners would look at this and go, okay, yeah. Something has deliberately hung this page and the browser.

But apparently this is a lucrative scheme, and it is being exploited whenever it can. And there's a history of this even before this particular API. The cretins have figured out how to lock up a browser after presenting this scary message in order to get people to call. And what happens is hopefully, or I guess not hopefully, but the convincing-sounding tech support person talks the person out of their credit card information in order to pay for the technical support required to fix this. And as we know, none of this is Microsoft behavior. So again, this is not the way Microsoft works.

Leo: I'll have a new answer for people who call. I mean, of course when people call saying I got this notice that I should call Windows about my bug, I usually tell them don't. But I didn't realize that they could freeze the browser. That's cool. I mean, that's not cool.

Steve: Yeah, that's not cool. But interesting.

Leo: But cool, yeah.

Steve: Yes, yes. So I just sort of also wanted to put out a reminder that it's worth auditing your app permissions from time to time. We've talked about this over on the Android platform. But there was a piece in the Wall Street Journal suggesting that, although Google had said they had stopped rifling through people's email for the sake of choosing ads and any kind of targeting, it's not the case if you have given apps permissions on your email.

There's a place you can go, myaccount.google.com/permissions, and I went there because I was curious. It turned out that my favorite PDF reader for iOS, iAnnotate, has access to my Google Drive, and I want it to. There were a bunch of things which were able to use Google for authentication, so login with Google. Then Chrome itself had full access to my Google account; and YouTube TV, which I've started to experiment with, has access to YouTube. But otherwise, I didn't have any things that looked worrisome.

However, the Hacker News covered this Wall Street Journal article and noted that, for example, an app like SaneBox, which is a third-party manager of a user's Google Mail inbox, would, as you'd expect...

Leo: Of course it would.

Steve: ...have full access to it.

Leo: So you have to tell somebody that?

Steve: I know. Although the point is that many times you may have used something for a week or two and then decided, eh, I really don't care about this anymore. So it's worth just going through and making sure there are no stale permissions that you're giving apps that no longer need them.

Leo: Anytime I read an anti-Google article in the Wall Street Journal I always have to judge whether it's a hit piece, which I judge this to be, because there's kind of a hereditary animosity between the Journal and Google because of course Google's stealing all their advertising. So you've got to read it with a grain of salt. Yeah, basically the point of the article is, if you give applications access to your Gmail, they can read your Gmail. Well, duh.

Steve: Well, it's also, though, that if you stop using applications, then...

Leo: Oh, you should turn it off, yes.

Steve: Yes. Apps don't remove their own permissions.

Leo: Right.

Steve: So it tends to be a cumulative thing. And from time to time just going through it, you know, you might well find, oh, what? I forgot that I had given that thing that I'm no longer using full access to my email.

Leo: It's a little like Louis in "Casablanca," though. "I'm shocked - shocked" - to learn that applications are reading my email. Of course they are. And then they kind of say, and by the way, it's not just computers, sometimes it's humans, because during the machine learning process humans are used to train programs. You know, let me tell you another thing you could be shocked by. If you have an antispam filter, if any of your web-based emails have spam filtering, which all do, they're also reading your email. There's no other way to filter spam.

Steve: Yup.

Leo: So, I mean, I don't know. I feel like the Wall Street Journal really was - this was link bait.

Steve: Yeah, still.

Leo: Yeah, I mean, it's worth absolutely noting, and certainly we should note, that you should go through those permissions all the time, not just for mail, but everything else, see what else. And on Twitter and on everything else, Facebook, that you give permissions to because you want to make sure that those people still deserve it.

Steve: Yup.

Leo: It's like saying "Windows reads the contents of your hard drive. I'm shocked."

Steve: No, it's not, but okay.

Leo: Well, it's kind of like that. These are all apps that you specifically signed up for, like TripIt or Unroll Me, to process your Gmail and tell you something. Well, there's no other way to do it. I think they're pretty clear when they say that we're going to look at your email. It's not like a human's doing it. All right. Back to you, Steve.

Steve: So we now have malware becoming a little picky about what it chooses to do when it gets into a person's computer. Kaspersky Labs discovered a new variant of the Rakhni, R-A-K-H-N-I, ransomware, although it's more than ransomware, which has been upgraded to include cryptocurrency mining also. But the malware faces a dilemma since mining cryptocurrency and encrypting all of the computer's valuable files tend to be mutually exclusive. You can't do both.

So what's a malicious infection to do? Well, it literally checks out the system that it's in. It first performs some anti-virtual machine and anti-sandbox checks to decide whether it should infect the system at all without being caught. It doesn't want to be analyzed, so it checks to see if it's running in a virtual environment to the best of its ability. If it looks like the coast is clear, it looks for a bitcoin folder in the app data section of the user's machine, which suggests to it that the system's owner may have something to lose from encryption. So it installs ransomware to essentially encrypt the user's wallet along with everything else. It terminates all other running processes that might have files locked or that might stop it, then proceeds to encrypt the user's files and displays a ransom note in a text file.

But if a bitcoin folder doesn't exist, and the machine is a bit beefier, with more than two logical processors, which would mean more than a single hyperthreaded core, it decides that it's worthwhile to do some mining. So instead it installs a cryptocurrency miner and gets to work doing that. Oh, and the cryptocurrency miner is something called MinerGate, which is a service that simultaneously mines Monero, Monero Original, and Dashcoin. MinerGate describes itself as "a mining pool created by a group of crypto coin enthusiasts." They said: "It's the first pool which provides service for merged mining. This means that while mining on our pool, you can mine different coins simultaneously without decreasing the hash rate for major coins."

Anyway, so while that's all happening - oh, and this is again further indication of the trend that I was noticing. It uses the Windows Cert Manager utility to install a root certificate, actually two certificates, one which claims to have been issued by Microsoft and the other by Adobe, in an attempt to disguise the miner as a trusted process. So it

pretends to have been signed by Microsoft, and it's able to carry that off, that spoof off by sticking a certificate in which is not actually a Microsoft certificate, but which does sign that piece of malware. So a user who is somewhat sophisticated would check the signature on the malware and say, oh, it's valid. It says the signature is valid, and it's signed by Microsoft.

Okay. If neither of those two conditions are met, that is, there's no bitcoin wallet, which would suggest that encrypting the drive would be beneficial to it, or if it's on a rather weak machine, it switches into worm mode and begins to scour the network for any attached machines where it's able to gain access and install copies of itself there. So anyway, here's another instance of certificates being installed in machines by malware, something I hope we're going to see our OSes become hardened to soon, and sort of the logical evolution of malware. Since it's now easy to do cryptocurrency mining, and it's also possible to do ransomware, well, what's the best strategy? So it decides on the fly, rather than only being a one-trick pony.

We've talked about Coinhive a lot. That's the service which it's sort of in the gray zone. They ostensibly offer a service, as we've discussed, where a website can decide that it wants to monetize its visitors' visits by forwarding Coinhive mining to people who visit that site to run a miner on their system. And after that caused a big ruckus, then Coinhive took the initiative of making sure that the user had given permission to mine on their system while they were visiting the site to keep bad guys from injecting Coinhive mining into other websites in order to mine maliciously, or at least in the background.

Okay. So the Coinhive guys are nothing if not innovative. Now they've created a link shortener service which incorporates proof of work. So you were just talking about a link shortener that you're enamored of, Leo, in MacBreak Weekly. Our listeners know that I like bit.ly, and I sometimes will use a bit.ly link tied to something short, just so our listeners can get to a long and confusing link easily. It's convenient. They're of course also a bit of a security concern because you can't see the link that you're going to without jumping through a little more hoops. So it can be used by bad guys to obfuscate a URL that you might not otherwise deliberately go to. Instead, you click the link, and it uses a browser redirect to take you to where you're really going.

So the Coinhive guys said, okay. Let's use browser mining to monetize link shorteners, or our link shortener. Coinhive states on their page where they're selling this service: "If you have a URL you'd like to forward your users to, you can create a [and then they have the domain] cnhv.co short link to it. The user has to solve a number of hashes which are adjustable by you and is then automatically forwarded to the target URL after they've done so." So, okay. Seems kind of hokey, but it would allow the monetization of link forwarding.

So unfortunately it didn't take bad guys long, much as they abused the previous offering, to figure out a way to abuse this one. Luke Leal of Sucuri blogged about this a couple months ago, and it's just come back onto people's radar because it's now becoming quite prevalent. There are hundreds of sites which are now generally various types of content management based. So they're using some sort of hacks to plant these links.

There is an obscured script which is being embedded on sites which, when turned back into text, that is, deobfuscated, will render an iframe which contains a 1x1 pixel window. I saw some pictures of it. You could easily, I mean, you'd have to look for it in order to find it. It's in the upper left-hand corner, a one-pixel iframe. And so it's of course doing the same thing. It's being planted on websites' delivery pages. People are as a consequence having their computer, after running through this link redirect, essentially back on Coinhive.com, running bitcoin miners on behalf of the people who planted the links on other websites.

So once again, the Coinhive people, I mean, again, they're being innovative. But it turns out that, when Luke posted his piece, no AV engines as reported by VirusTotal were objecting to this little URL that's being planted. I checked last night, and nine of 67 different AV tools were listing it as dangerous. So I expect that more will be following. And in fact I just clicked on it now to see how quickly this is going up, and VirusTotal says, nope, still nine out of 67. Remember that we talked about VirusTotal last week as something you could upload malware to. You can also give it suspicious URLs and see whether there's anything that anybody objects to.

So what's going to happen is that the AV tools will block this, and web browsers just are overall, I mean, I guess right now we probably have bit mining blocking for users who have deliberately installed them in web browsers. I presume a lot of our listeners have because we've been talking about this, and this has been a thing now for months. But there are certainly lots of people who don't yet have their browsers blocking through the addition of an add-on, and browsers are not yet doing so natively. We need to have that happen, and then this whole annoyance will go away.

Leo: This site was not a URL - you'd actually, if you don't mind a little sidebar, be interested...

Steve: Yeah, yeah, yeah, please, please.

Leo: It's itty.bitty.site. But it's not a URL.

Steve: Oh, your site.

Leo: The one I was talking about.

Steve: Yeah.

Leo: Because what it does is it base64 encodes whatever you put on the site into the URL. So the URL, which is long as a result, would work offline. Your browser renders it. So it allows you to do about a page of text. You could even do, if you look at the itty.bitty.site website, you could even - and this is the potential risk, same risk as with any site because you could put a script in it. You could actually do an app in this thing. Here's a little calculator.

Steve: That's very cool.

Leo: That's completely encoded in the URL. Which is amazing. We can even generate - in fact, I'll show you this one. If anybody trusts me, this is "Jabberwocky." I could do the QR code for "Jabberwocky." And if you scanned the QR code, it would actually show you...

Steve: Take your browser to that location.

Leo: Yeah. Without being online. So if you scan this with your phone, you'll get a page that has the Jabberwocky in it, but without being online because it's entirely encoded base64 into the URL. Which is wild.

Steve: Very cool.

Leo: Yeah. And of course potentially hugely dangerous. I would never mention that to anybody unless they knew that.

Steve: I wonder if they compress the ASCII.

Leo: They do. They use Lempel-Ziv.

Steve: Ah, okay. Yup, that's exactly what I would do. I would compress it and then base64 encode the binary.

Leo: Exactly what they do. You are so smart. It's kind of a neat hack, really.

Steve: Yeah, it is, yeah. It really is. So we all know D-Link. They're one of the major consumer router and little switch providers. I've got D-Link stuff around. Unfortunately, as can happen, but really should not happen, they lost control of their own code-signing certificate, which is really bad. Security researchers from ESET recently identified two malware families that were previously associated with the cyberespionage group BlackTech. And that malware had been or has been signed using valid digital certificates belonging to D-Link and another reputable Taiwanese manufacturer known as Changing Information Technology. Not as well known to us in the U.S., but presumably in Taiwan.

The first malware, named Plead, is a remotely controlled backdoor designed to steal confidential documents and to spy on users. The second piece of malware is a password stealer designed to collect saved passwords from users' browsers - Chrome, IE, Microsoft Outlook, and Firefox. So also apps in addition to browsers. So the researcher notified D-Link and Changing Information Technology, both of these companies, about the issue, and the companies immediately revoked the compromised digital certificates on the 3rd and 4th, so last Tuesday and Wednesday of last week.

However, unfortunately, as we know, certificate revocation relies upon the entity trusting the certificate's identity assertion to deliberately and continuously check with, or at least the first time it's encountered for that particular use, to check with a certificate's issuer and signer, that is, the CA that issued the certificate, to reverify the certificate's present real-time validity. Okay. So, right? Certainly lots of D-Link routers have firmware previously and legitimately signed with their cert. So if the router, as I hope it would, would check that the firmware has been signed by D-Link, it would be nice if it also checked for revocation.

But almost nobody actually does that in practice. It takes time. And just it's not something that is often enough done. And it turns out that indeed most antivirus software does not bother to check the certificate's validity. The fact that it is a valid certificate signed by somebody in the root store and still within the validity dates, or - and this is interesting about certificates which sign code - those certificates contain a co-signature, that is, it's countersigned with a timestamp which is obtained at the time of

the signing. And that's important because, as we know, certificates themselves have a multiyear life, but like only three or four or five years. I think code-signing certs have a longer life, but not forever. But the code could live on on a repository somewhere.

So the code itself could be signed with a certificate which has since expired, yet it was valid at the time of signing. So in order to know that the certificate wasn't signed with an expired certificate, the act of signing can bind a timestamp into the certificate which cannot be spoofed because it comes from a signing authority, a timestamping service, so that's bound in. Okay. So what this means is that the certificate's been revoked, and we need to honor - we need to continue to honor signings by that code-signing certificate which were signed as indicated by the timestamp previous to the revocation time.

Now, the BlackTech hackers were using a non-revoked certificate to sign their certificates. So as long as they did that and got them timestamped, as they presumably did, revocation after that fact would not invalidate the certificate. So this is a mess. The D-Link certificate in question was issued by Symantec on September 29th, 2016, so not quite two years ago. And the certificate is valid for more than a year from now, that is, until September 30th, 2019. So this particular certificate had a three-year life. After this date, that is, after the certificate's actual expiration date, new signings under the certificate will be no longer valid. That is, the certificate itself would be invalid independent of revocation checks. But until then, that is, until a year from now, more than a year from now, any new malware that is signed by this certificate, unless the AV checks revocation of all certificates, is going to be deemed valid.

So again, as I said, this is a mess. What will probably have to happen is that this particular certificate's thumbprint will be kind of reverse-pinned. It'll be blacklisted by thumbprint, which cannot be spoofed, as we know, in responsible AV vendors and maybe even hopefully in operating systems will no longer trust things that have been signed after the date that D-Link says they know that the cert escaped their control, and if any software has been signed with that thumbprint. So it's a mess when code-signing certs escape because, even when the system, I mean, first of all, the system is not working as well as it should because that would say revocation would always be checked. In practice, that doesn't happen.

So the best we can do, in terms of practical use of the system, is for notification to happen and for operating systems and antivirus vendors maybe who are able to respond more quickly. I mean, they're desperate, AV vendors are, to continue demonstrating that they're adding value as operating systems continue to take on more responsibility for protecting users natively. So not good when this happens. And especially from a major company like D-Link. I'm sure they're not happy that the cert got away from them.

Okay. iOS v11.4.1. This happened just last night. I updated a couple of my iDevices today so that I could see this and experience this for myself. There was a feature which first appeared in the iOS 12 beta. And it wasn't clear whether Apple would hold the feature back for iOS 12. We learned officially last night that they decided not to. There's now a switch which has been added to the Face ID or Touch ID and Passcode section of the Settings app which allows for the essentially USB access to the phone to be disabled when the phone is locked. So that's a cool new feature. The caption under the feature explains that you can unlock the iPhone - it explains: "Unlock iPhone to allow USB accessories to connect when it's been more than an hour since your iPhone was locked."

So what happens is you have an hour after the iPhone is locked before a timer expires which then suspends USB support until the phone has been unlocked. Sounds like a nice feature. What is believed is that this was Apple responding to the Lightning connector-based hacks which are used now to jailbreak or by law enforcement to get into phones. Immediately after the upgrade, which is to say yesterday, ElcomSoft - which is a company we've spoken of a number of times. They're in this business of messing with

iPhones. They confirmed and blogged in detail about a workaround which they had seen previously in the betas and which did survive in the final release result.

So no one's really exactly sure what's going on here. It's a little complicated, so I'm going to quote directly from a couple paragraphs of their blog. They said: "The most spoken about thing about iOS 11.4.1 is undoubtedly USB Restricted Mode. This highly controversial feature is apparently built in response to threats created by passcode-cracking solutions such as those made by Cellebrite and GrayKey on unmanaged devices. The new default behavior is to disable data connectivity of the Lightning connector after one hour since the device was last unlocked, or one hour since the device has been disconnected from a trusted USB accessory. In addition, users can quickly disable the USB port manually by following the SOS mode routine." That's the click the power or sleep button five times rapidly.

They write: "Once USB Restricted Mode is engaged on a device, no data communications occur over the Lightning port. A connected computer or accessory will not detect a smart device. If anything, an iPhone in USB Restricted Mode acts as a dumb battery pack. It can be charged, but cannot be identified as a smart device. This," they write, "effectively blocks forensic tools from being able to crack passcodes if the iPhone spent more than one hour locked. Since law enforcement needs time (more than one hour) to transport the seized device to a lab, and then more time to obtain an extraction warrant, USB Restricted Mode seems well designed to block this scenario." And they say: "Or is it?"

They write: "We performed several tests and can now confirm that USB Restricted Mode is maintained through reboots, and persists software restores via Recovery mode." So basically they played with it, trying all kinds of different tricks, seeing if they could regress out of restricted mode. They said: "In other words, we have found no obvious way to break USB Restricted Mode once it is already engaged. However, we discovered a workaround, which happens to work exactly as we suggested back in May."

Okay. So they report that, if any Lightning device is plugged into a locked iPhone any time before the 60-minute timer has expired, whether or not the device is known to and trusted by the iPhone, the lock countdown timer will be restarted for another 60 minutes, and this can continue indefinitely. Then I've skipped a bunch of stuff in the blog.

But they conclude saying: "We've seen rumors about GrayKey being able to defeat protection provided by USB Restricted Mode. At this time, these are nothing more than rumors. The company's official policy is never issuing comments about pre-release software. With iOS 11.4.1 just released, we'll have to wait to see if the new security measure can be defeated. Either way, since iOS 11.4, the speed of GrayKey, and probably its competitors" - this is ElcomSoft writing - "is limited to slow recovery rates of one passcode guess every 10 minutes."

They say: "While this allows breaking four-digit passcodes in reasonable time," they say, "about two months worst-case scenario, six-digit passcodes already make little sense to attack unless one has a custom dictionary; and, as we know, six digits is the new default length for the passcode suggested by iOS."

So anyway, an interesting new feature. iOS moves forward, continuing to work to improve the security of their users. And after an hour nobody sticking something into the Lightning port will be able to access it. And Leo, how does this comport with what Rene explained?

Leo: Well, we read the ElcomSoft article.

Steve: Oh, okay.

Leo: And as you pointed out, ElcomSoft had some interest in saying, oh, no, no. But basically we concluded was, if law enforcement wants to carry - ElcomSoft used as an example the Apple Camera Connection Kit. But any USB Lightning port device is designed to keep the Lightning port alive, even after locking. And that's the key; right?

Steve: Right.

Leo: Apple's decided, well, if we were to lock it, and you're copying photos off on your camera to the iPhone or iPad, and then it locks and suddenly stops working, you'd be peeved.

Steve: Right.

Leo: So that keeps, even though the device is locked, it keeps the USB channel open. So we said, well, if you're law enforcement, what you do is you carry USB, the Camera Connection Kit or something similar. And you apprehend somebody, you grab his phone, presume that he has unlocked it sometime in the last hour, plug this thing in, and then Rene pointed out you probably also want to put it in a Faraday bag so that...

Steve: Oh, right, right, right.

Leo: ...it can't be remote wiped; right?

Steve: Yup, yup.

Leo: Put it in a Faraday bag and then take it back to the home office. You've kept the USB port alive.

Steve: Got it.

Leo: And nobody denies that that's the case. That is the case. So, and there's a good reason for it, and maybe Apple will respond. But, you know.

Steve: Yeah. It sounds like they're aware of it, and it was a tradeoff they had to make. And, first of all, again, our listeners should know, if you click the power button five times rapidly, that immediately locks the USB. It puts it into USB Protected Mode. So that's protection. Or as you said, if it's been more than an hour, then it's already locked.

Leo: Right, yeah. I think this is a fairly good idea.

Steve: Yeah. Oh, I agree. Again, Apple doing the right thing for us.

Leo: So if I press my phone - if I press this five times, one two three four five, and then do what? Just leave it?

Steve: Yeah, exactly. That is now - and I don't know if you have to power it down, or if that's enough. I couldn't get an...

Leo: Well, once you've powered it down, you're screwed, you know, that's the whole thing; right? Now they can't win at all.

Steve: Right. Right, right, right.

Leo: And this is the thing we also talked about. If the data on the device is encrypted, even if it's locked, even if they have one of those devices plugged into it, they're going to pull off encrypted data, not unencrypted data, once the phone is locked; is that right?

Steve: Correct. Correct. Well, what I thought was interesting was that it can, if they can get to a live Lightning port, these crackers can do an automated guess once every 10 minutes without triggering the phone's "you've guessed too many times" timer. So what they're doing is they're just, like, sticking the phone on this cracking device and waiting maybe up to two months if the user has a four-digit passcode, or hoping they get lucky if it's longer than that.

Leo: If it's longer. And of course the best solution, which is something I do, Andy does, a lot of people do is don't use a passcode at all. Use a long - I have an 18- or 19-character password. It's a pain in the butt to enter it. But then I don't have to worry at all because no brute force is going to [crosstalk].

Steve: Yup, [crosstalk] because what this does is it dramatically restricts the rate at which brute forcing guessing can happen.

Leo: Right.

Steve: Just another quickie, another reminder. I sort of thought it was interesting. As we all know by now, ever since Peter Norton made a name for himself with his famous Undelete utility, you know, that was the one thing Peter came out with that, like, put him on the map.

Leo: And it was pretty simple thing, too.

Steve: Oh, my god, Leo.

Leo: He wrote it in Pascal.

Steve: Yeah. And it turns out that the first character of the filename was turned into an E5, a hex E5, just a little bit of trivia that I remember.

Leo: It looks like an upside-down E.

Steve: Yeah.

Leo: If you look at it as an ASCII code.

Steve: Yup, yup. And so what he realized was, oh. Oh, and the FAT was - the entries in the File Allocation Table were also zeroed to free up the space.

Leo: Right.

Steve: But of course, as we know, the data itself was not proactively removed from the hard drive. And so, if you were lucky, you could restore the directory entry which was left there intact, pointing to the first cluster of the file. And often the file was a contiguous run of clusters. The directory entry also showed how long the file was. That told Peter's Unerase how many clusters it needed. And so it was able to judge whether there was a contiguous run of clusters that long. In which case, chances were you had the file back. And so that was, whoo, that made Peter famous and so forth.

Anyway, as we know, deleting files doesn't delete them. Neither does formatting them. Formatting is a stronger process. It wipes out the front of the drive. The whole directory, the FAT, the File Allocation Table gets zeroed and so forth. But the data, again, is still out there on the physical drive surface. Well, this of course is the same with memory cards, which use the same sort of file system. So Bleeping Computer just carried an interesting little piece from researchers at the University of Hertfordshire in the U.K. They purchased a bunch of random memory cards from eBay, from some auctions, from second-hand shops, and other sources over a four-month period.

They discovered, that shouldn't come as a surprise to us, that two thirds of memory cards contained readily recoverable data. They wrote in their report that the memory cards they recovered were previously used in smartphones, tablets, cameras, satellite navigation systems, and even drones. To perform the recovery they first imaged the storage to create a bit-by-bit copy, then just used freely available software to see whether they could recover any data from the card. The team recovered, they wrote, a great deal of data from the memory cards: intimate photos, selfies, passport copies, contact lists, navigation files, pornography, rsums...

Leo: I'm going to have to erase my cards better.

Steve: ...browsing history, identification numbers, and personal documents.

Leo: Of course. You put all that stuff on USB cards.

Steve: Yes, yes. And what you want to do is never, ever, ever throw them away.

Leo: Is there any other way to fix that?

Steve: Well, I'll talk about that in a second. Thirty-six were not wiped at all, so neither the original owner nor the seller took any steps to remove the data.

Leo: Didn't bother.

Steve: Yeah. Twenty-nine appear to have been formatted, but the data was still there, so it was recoverable with minimal effort. Two cards had data deleted, but was still easily recoverable. Twenty-five appeared to have been wiped using a data erasing tool that overwrote the storage area. So from those cards nothing could be recovered. Four were dead completely and could not be accessed. They were just broken. And then four had no data present, but the team was unable to determine the reason.

So there are, and it is worth using, data wiping tools. I've mentioned previously on this podcast that I have an arc of my future development. As we all know, I'll be working on SpinRite 6.1 and .2, which will dramatically speed up SpinRite. That's like the main feature is speed, by talking directly to the hardware, first for the motherboard-based connected controllers, the AHCI, SATA, and IDE, if anyone has an older motherboard, basically circumventing the BIOS. Then I'm going to do the same thing for the serial connected devices to give us good performance on attached USB. And of course that also means thumb drives and smartcards.

So what I'm going to do before I start on 7 is to release another product called Beyond Recall. And as its name suggests, whereas SpinRite is all about recovering data, Beyond Recall will be about absolutely and positively and quickly - because what it's going to do is, the idea is it's going to use the technology I develop for the 6.x series. I will reuse that low-level direct-to-the-hardware technology to make Beyond Recall feasibly fast. So, for example, you'll be able to wipe a multi-terabyte drive in a few hours; whereas it's just prohibitively, I mean, it's like incredibly painfully difficult to do that in a reasonable time otherwise.

Leo: Or you could do what Mr. Robot did. Throw them in the microwave.

Steve: Yes, yes.

Leo: Would that work?

Steve: Well, if you have a throwaway microwave. Because, I mean, it hurts the microwave.

Leo: It's not good for microwaves.

Steve: It creates sparks and all kinds of sh*t. I would drill holes in them. I mean, you really want to physically destroy them until I have Beyond Recall, and maybe even before, or after. But anyway, so I think there's an increasing need to securely wipe these things, and there's also sort of lots of tricks I'll get up to because, by talking to their controllers, you can deal with things like regions which have been swapped out and are no longer accessible to the front door. You need to get in the back door in order to deal with regions which have been taken out because of wear, you know, the whole wear-leveling thing. So anyway, that's going to be...

Leo: That's what we really need because there's always that stray stuff lying around because of wear leveling.

Steve: Yup, yup, yup. And lastly, just to close the loop on the USB fans which were handed out at the Trump/Kim summit, many security researchers were made very uncomfortable by this. Everybody got them, and two researchers were able to obtain them from journalists who were present.

Leo: Oh, good. Oh, good.

Steve: Yes.

Leo: Was there anything going on there?

Steve: No.

Leo: Oh.

Steve: They found nothing other than fan motors hooked directly to the USB's 5V power and ground lines. The two data lines had no connections to them, so nothing smart was there that might get up to some mischief. However, the security researchers noted that, first of all, many of these - noted that many fans were distributed, and a few smart fans might still have been used in targeted attacks, that is, given to specific individuals who would then maybe take some comfort from knowing that security researchers opened other people's fans and found nothing in them. So the overall takeaway, not only from this but in general, is do not use untrusted USB without a USB condom. It is always too dangerous.

Leo: Which thanks to you I always have with me.

Steve: Yup.

Leo: Steve sent me two condoms.

Steve: Yup. And Leo, these are reusable condoms because, you know, you should really never reuse...

Leo: Never use anything else, yeah.

Steve: ...the other kind. Anyone traveling, especially anyone who might be targeted, should always use protection. Or better yet, simply abstain while on the road.

Leo: Yes.

Steve: However, if you can't...

Leo: Sometimes you have to charge; right?

Steve: It's worth having a condom.

Leo: You know, I'm sure the Secret Service vetted, I would hope, vetted them.

Steve: As I understand it, I mean, I don't know how they did. You could put a - remember there were those USB killing devices we talked about a while ago that would, like, zap a USB - it was a gadget that you would hook to your computer, and it would charge up and then release a kilovolt burst into the USB port.

Leo: Oh, geez. Oh, that'd be handy.

Steve: So one thing you could do would be to just blast the crap out of the data lines, which would fry anything that was sniffing the data lines, but still leave the power lines alone. So anyway...

Leo: This is the PortaPow.

Steve: That's the one. PortaPow.

Leo: Data block plus smart charge that you just use as a little USB condom.

Steve: Yeah. And it just - all it is, is the 5V and ground lines go through, and the data lines are disconnected.

Leo: Yup. And you're safe. No problem.

Steve: So I got an interesting note from Eric, looks like - boy, I didn't pronounce his name ahead of time. I didn't practice. Theriault, T-H-E-R-I-A-U-L-T, Eric Theriault. I hope I didn't mangle...

Leo: Theriault. It's French.

Steve: So it's pronounced...

Leo: Theriault, I would guess.

Steve: Okay, I'm going to give up.

Leo: Yeah, Theriault.

Steve: Anyway, he's in Quebec. And is that Quebec or...

Leo: Quebec.

Steve: I don't know what I'm doing. Anyway, I do know what I'm doing about SpinRite. So his subject was "SpinRite & RAID 6: How to, please help." And he said hi to everybody. He said, "Hi Steve, Leo, Sue, Greg, and Elaine." So he missed some of your staff, Leo, but he got all of mine.

Leo: Close enough, yeah.

Steve: He says: "I have a home server set up on a RAID 6, five disks of 2TB." Each, I assume he means. "This computer is always on, and multiple services depend on it. I can easily shut down and reboot, if it's for a short period; but I would prefer not longer than 30 to 60 minutes. Worst case would be to shut down during the night.

"My question: How, if there's a way, can I do SpinRite maintenance on all five disks without having to shut down the computer for a week? I could do a 'hot unplug' one disk at a time and SpinRite it on another computer. But if my understanding is right, this would be useless" - well, no, but okay, we'll see where he goes with this - "because it would then make the data on that disk irrelevant. And when I would reconnect the drive, the software RAID would have to rebuild all the data on the reinserted drive because the data in the still active four disks would of course have changed." Okay, yes, he's right about that.

"Please share, as usual, any good practice for this kind of configuration, on or off the podcast, up to you." Well, thanks, Eric. He says: "FYI, this is a software RAID on a Linux using MDADM." Which I'm not familiar with, but sounds like a nice thing. He says: "Oh, by the way, I got in the show first back in July 2017" - so just exactly a year ago, he says - "and after three months decided to go back from the beginning. I'm now up to the 217th episode and, of course, up to date since July last year." So he's going back and catching up. So yay, Eric, thanks. He says: "Keep going with that amazing podcast. I learn so much every week. Thank you so much."

Okay. So I thought about his problem. And the best I can suggest, it would be to take advantage of 6.1's speed when it's available. Assuming they're five disks of 2TB each, rather than a total RAID 6 of 2TB, which would make all the drives maybe a half a terabyte, before I suspended the work on 6.1 in order to work on SQRL, I had

benchmarked the preliminary driver that I wrote for AHCI controllers at half a terabyte per hour, which would mean that SpinRite could do a 2TB drive in four hours, making it at least feasible to do overnight.

So what Eric could do would be to down his system. And presumably, if it's a Linux RAID, it's on a machine, so on a PC, so he wouldn't have to remove any drives, just run SpinRite on one of those drives. And actually we talked about the command line option. You could use a script to run it successively on one drive, then a second drive, which would be two drives overnight in eight hours, then bring it back up. So you could, in this scenario, you could do the whole five-disk RAID in three nights in a way that would not break the RAID because SpinRite never changes the data on any of the drives it works on. So the RAID wouldn't break, and in three nights you'd have SpinRited all of the drives.

And nothing I will ever be able to do will be able to improve on half a terabyte per hour. That is absolutely, I pulled every trick in the book. Even though I'm in DOS, I'm using a fancy real mode that gives access to extended memory from real mode, so I'm allocating a 64MB buffer, which is the largest transfer that any drives can do in a single burst. And all drives are optimized for continuous streaming single-burst transfer. So no revs are missed. It's like catching every single revolution. So we're limited by literally the platter speed.

Now, it might be that you have a higher platter speed, more sectors per, so that would increase its throughput. But basically, when we get to 6.1, it will be running the drives at absolutely their maximum capacity, like never missing a revolution in a 64MB transfer of, what, 128,000 sectors or something? Wow. Anyway, whatever it is, that's my target for 6.1. And as I mentioned before, Beyond Recall similarly will be employing the same technology to scrub disks.

I asked, when I was working on 6.1, I had suggested to the gang in the newsgroup where we work on this stuff that I add a Beyond Recall feature to SpinRite, and they universally said no, no, no, no, no. Do not confuse SpinRite...

Leo: That's probably a good idea.

Steve: Do not confuse SpinRite that recovers data with something that is called Beyond Recall that absolutely destroys the ability to recover. So it's like, okay, fine. That'll be a separate inexpensive product. And I think it'll be popular. So we'll see.

Leo: There is Darik's Boot and Nuke, but that doesn't know about SSDs; right? That's probably the problem.

Steve: No, DBAN, yes, exactly.

Leo: DBAN, yeah.

Steve: So it'll be - I will have something as soon as I can.

Leo: Nice. Until there's DBAN. Until there's Secure - what is it called? Secure Erase? Total...

Steve: Beyond Recall.

Leo: Beyond Recall. I love that name.

Steve: Isn't that great? Yeah.

Leo: Until there's Beyond Recall, there's DBAN. But once Beyond Recall comes out, Darik, you're going to have to find a new line of work. So, no, he doesn't charge for it, so it's not really a line of work.

Steve: Okay. So STARTTLS. I really do want to do an extra "T" in there, Leo, every time I see it. START TTLS. No.

Leo: No.

Steve: I guess it's because of TTL, you know, Time To Live.

Leo: Right, right.

Steve: And so I'm used to thinking about that instead of TLS. So anyway, STARTTLS Everywhere is the name that the EFF, the Electronic Frontier Foundation, gave to their newest initiative. Once upon a time, just like the web and HTTP, email was only plaintext. Web pages went to and from in plaintext, and so did our mail. With the web, however, we have a direct point-to-point real-time link between our web browser and the remote server that we're visiting. So this point-to-point link made it a relatively simple thing to allow the remote web server to assert its identity and for the user's web browser client to verify that identity and to complain to us in real-time, you know, with warnings and stuff, if something didn't look right somewhere.

But email has always been different. For one thing, it's designed to be non-real-time and asynchronous. It's very much like the way packets route around the Internet that we've talked about. Email is delivered as a best-effort store-and-forward system where autonomous SMTP, that's Simple Mail Transfer Protocol servers, receive and forward email toward its destination. And in that sense store-and-forward is a bit like the Internet router's receive-and-route function, where the exact path to be taken may not be known, and so the various components along the way just sort of send the mail toward its destination.

So to do that in the beginning we had, as we know, two elder email protocols, still in use today, although one has sort of faded a bit, POP, the Post Office Protocol, and SMTP, Simple Mail Transfer Protocol. POP is and was used by mail clients to receive mail from servers, from their so-called Post Office. The clients would connect to their email server over port 110 and ask for any email that had arrived since their last check-in. Typically they'd receive, then delete, the email from the server so that the email took its final hop to them and then had arrived at its destination. And no encryption, just plaintext, and we were all happy.

Then later IMAP came along, the Internet Message Access Protocol. And the way it's different is sort of hinted at by its name. It operates over port 143, and it allowed clients to retain their email on the server and instead manage it remotely through their client, which became more of a viewer of the email on the server. And again, port 143, no encryption anywhere. No encryption in sight.

And then finally the SMTP, the Simple Mail Transfer Protocol, that's the incoming mail service and the protocol which is offered by servers listening for connections on port 25. So whereas both ports 110 for POP and 143 for IMAP were getting mail from servers, port 25 is sending mail to servers. Whether it's the client sending it to their own SMTP server for it to then in turn send to the destination, or for one SMTP server to send it to the next SMTP server. And again, no encryption from the beginning.

And so even today we have email hopping around all over the place, and end-to-end encryption is far from guaranteed. Remember that you may have a secure connection to your SMTP server. But it's increasingly the case that the SMTP server you have sent your mail to directly connects to the SMTP server your recipient uses. And so if that link is secure, you're probably okay. But the protocol doesn't guarantee that. It could be multiple hops. And in that case the intermediate SMTP server is like a man in the middle. Even if there are secure connections on both sides of it, it's decrypted. That email is decrypted while it has it. And even when everything is working with email encryption, that will still be true. So it's only by the user taking the initiative to use PGP or S/MIME or to pre-encrypt a blob and then send it, that they can actually get end-to-end encryption. Even once everything is working with email encryption, it still isn't true end to end.

So how do we fix where we are? Well, unfortunately, it wasn't a straight line. Just as the first web browsers originally connected, as I mentioned before, over port 80, without any authentication and encryption, what the architects of the Internet did was they said, okay, we're going to give web servers a new port, a different port, in this case 443, which will require encryption. That is, anybody connecting to port 443 is still connecting to the web server, but has to negotiate in SSL originally and now a TLS protocol connection, meaning that the server will assume that and will send a certificate to assert its identity. The client will obtain it, make sure that it's the server it wants to talk to. They negotiate, and off they go, but all of that as a consequence of connecting on this different, dedicated port.

So the same thing was done for email. But unfortunately, it didn't happen first. It sort of happened after the - it's, like, not really clear. I've attempted to sort of like disentangle the RFCs, but it's a mess of one obsoleting the other. And I don't know, I can't really explain how we got into the mess we're in today because it seems unlike the orderly process we're used to on the Internet. But what happened is there exists today similar dedicated encryption-only ports for these three protocols, for POP, IMAP, and SMTP.

For POP, which used to be and still is unencrypted by default, there is POP over port 995. And any client connecting to port 995 of an email server is expected to bring up a TLS connection just like a web browser connecting to a web server on port 443. Certificate is exchanged. Certificate is verified, authenticated. Encryption comes up, and we're off and running. IMAP's original port 443 similarly has a twin, in this case with implicit encryption. That's port 993. And similarly, port 25 that is plaintext SMTP now has an encrypted version 465.

Okay. So we had new ports that were encrypted. Unfortunately, somewhere along the way, before this right way of doing it happened, STARTTLS was mixed in. But it's not quite that simple, either. The Internet wizards disliked the idea that port 25 and the encrypted version 465 was being used for both email forwarding between SMTP servers, also known as MTAs, Message Transfer Agents, but that same port was being used to receive email submissions from email clients.

So they further complicated things by attempting to split the submission and the forwarding roles by defining another SMTP port, 567, for email clients to use. So this was defined for some reason as a plaintext port which could be upgraded by the use of STARTTLS, which we'll get to in a second. And then, believe it or not, after adding this new SMTP alternative connection port for clients, just this year, earlier in 2018, the decision was made to discourage the use of port 567 for future email client submission in favor of returning to using SMTP over TLS 465 and sharing that single email submission port between clients and servers. So it's a mess.

Okay. So these secure ports are now widely adopted, but not universally adopted. They use our well-known and well-proven existing CA-issued server domain certificates to work with these new ports, just as they work so well with HTTPS. The problem is that an alternative sort of half-baked solution was also added to the RFCs which allows a negotiation between non-secured connections to elevate itself. And so what we have today kind of works, but not very well.

As its name suggests, STARTTLS provides a means for, as I said, an insecure plaintext connection to start using TLS. When a client connects - and this is over the non-implicit TLS ports, that is, not 995, 933, and 465, but over the old port, the old school, the original 110, 143, and 25. When a client connects, the server will respond with a hello message which includes the notification that it supports STARTTLS if it does. In which case, if the client also supports STARTTLS, it will accept that offer, essentially, from the server to switch over to TLS, and they will negotiate a TLS connection and switch the existing connection over a non-secure port to be secure. Sounds great; right?

Except, and again, this was like there was a point in time where this must have made sense, where the idea was to provide an upgrade path on existing ports at the protocol level that wouldn't break old stuff, that wouldn't require new ports to be assigned, that would sort of be forward-compatible. But, boy, in retrospect it's a mess. Yet it's in the world. We have it. And so we're sort of stuck with it.

Okay. So STARTTLS itself currently sits with about just shy of 90 percent, about 89 percent presence currently, as is shown by Google's email transparency report, which is not bad considering that it was at 39 percent just five years ago. So with the move toward security and a heightened concern for encryption, new servers coming online look like, well, they're probably just, you know, they default to supporting it because why not. You've got a certificate for the server. Let the email server use it, and you've got TLS. Except big problems.

Okay. So now paraphrasing from the EFF because they summarized this nicely, they said: "Although many mail servers enable STARTTLS [get this] most still do not validate the certificates. Without certificate validation [as we know] an active attacker on the network can read and modify emails sent through supposedly secure connections. Since it's not common practice to validate certificates, there's often little incentive to present valid certificates in the first place." Right? Certificates are a pain. Which is why a lot of sites, a lot of websites just didn't bother. And you could argue that email needs them even less than web servers do, where you actually do have sensitive data passing back and forth on a web server's pages. Yeah, email has lots of sensitive data, too. But somehow it's sort of out of sight.

Okay. But get this. Servers are not validating the certificates. So what are they doing? They are using self-signed certificates that no one trusts. The recipient is not going to trust a self-signed certificate. But it's valid. So, I mean, it is a certificate. So I guess the way to think of this is you would have opportunistic encryption, that is, the server is offering a certificate that does allow the TLS handshake to happen. So even though the authentication is not present, at least you get protection from a passive observer. You get encryption. So somebody sniffing the line won't see anything.

So, okay. The EFF writes: "On the web, when browsers encounter certificate errors, these errors can be and are immediately communicated to the end user" - as I mentioned before - "who can then decide whether to continue to the insecure site" or not. Or that is to say, the nonsecured site. "With email," they write, "this is not an option, since an email user's client, like Thunderbird or a Gmail app on a user's phone, runs separately from the machine responsible for actually sending the mail." That is, because it's this SMT server to SMTP server connection.

"Since breakage," they write, "means the email simply won't send, the email ecosystem is naturally more risk-averse than the browser ecosystem when it comes to breakages. As a result," the EFF writes, "the ecosystem is stuck with a sort of chicken-and-egg problem: no one validates certificates because the other party often doesn't have a valid one. And," as they write, "the long tail of mail servers continue to use invalid certificates because no one is validating them anyway." So why bother?

And there's more. "Let's imagine," okay, they write, "a future where everyone did have STARTTLS enabled and did have a valid certificate. So now it becomes safe for everyone to start to validate certificates and insist upon validation. What could go wrong?"

Well, we still have the classic security downgrade attack. Both communicating mail servers support STARTTLS. And they both have valid certificates. I mean, everybody's valid. And certificates are being checked; right? Unfortunately, the initially insecure connection is opportunistically upgraded to a secure one. So as we know, a man in the middle could intercept the traffic and remove the declaration of STARTTLS support from the connection-receiving server, who is the first one to say, hey, I support TLS. If you do, let's go. So as a consequence, the client believes the server it's connecting to, lamely, doesn't support STARTTLS. And so it shrugs and says okay, fine, we'll use non-encrypted connection. And so a classic security downgrade attack.

It turns out that this is not just theoretical. The EFF writes that U.S. ISPs and those abroad have been caught doing exactly this. And in 2014, four years ago, several researchers found that encryption on outbound email from several countries was being regularly stripped. So again, unfortunately, if you can, that's what happens.

Now there's DANE. DANE we've talked about. Everyone knows I am super excited about the idea of DNS-based Authentication of Named Entities. That's the idea of using DNS as a global caching sort of universal reference system where this, for example, could be used to solve the problem completely. If we had consistent and full DANE deployment, that would offer a scalable solution for mail servers which could clarify the certificate validation rules and prevent downgrade attacks simply by publishing in their DNS what it is they support for incoming email connections. In that case, the connecting server could - it would have pulled DNS anyway in order to get the IP address it needs to connect to.

So in the process it could ask for the DANE record for email authentication; find out, oh, look, this server does support STARTTLS, so we're good to go. In that case, if the server appeared to say "I don't support it," then the connecting server would say, "Whoops, something's wrong here. We're not going to send email over an insecure connection that could be secured." The bad news is DANE relies upon DNSSEC. And DNSSEC deployment stagnated, as been stagnant for about the past five years and is stuck somewhere between 10 and 15 percent globally. So that's not going to come to our rescue anytime soon.

Okay. So fresh off their success with Let's Encrypt, the EFF has set itself three worthy goals to improve this current sad state of affairs. First, they want to improve STARTTLS adoption. They want to make it easy to deploy STARTTLS with valid certificates, not self-signed certificates, on mail servers. So they are developing Certbot, as they call them,

certificate bots, Certbot plugins for popular mail server software, starting with Postfix, which is currently in beta.

So the idea would be that this would allow Postfix, much like Let's Encrypt allows a web server to ask for a valid trusted certificate from the Let's Encrypt CA, this Certbot would allow web servers similarly to obtain a no-cost mail certificate, like automatically. So certificates no longer need to be self-signed. They can be valid. They can be free. So then the EFF says that this first work for Postfix will be followed up by support for the Dovecot and Sendmail servers, and they are open to and welcome contributions of installer plugins for other email servers.

Okay. So improve the adoption, make it easy for servers to have valid certificates, and not cost them anything. Second, prevent STARTTLS downgrade attacks. They say that, in order to detect attacks, they (the EFF) will be hosting a web server policy list of mail servers known to support STARTTLS. This list will act as a preload list of mail server security policies. The list already includes a number of big player email domains - of course all the usual suspects, Gmail, Yahoo, Outlook and so forth - and they're actively seeking more. Use of the list will require mail servers to become list aware, of course; but without this STARTTLS will always be susceptible to downgrade attack. You know, I have to say my feeling is it would be better just to switch over to the TLS implicit ports and just give up on STARTTLS. But I certainly respect what the EFF is doing with this effort.

And, finally, the last part of this initiative is lowering the barriers to entry for running a dedicated mail server. And I was a little puzzled by this one. They write, when they're explaining this: "Email was designed as a federated and decentralized communication protocol. Since then, the ecosystem has centralized dramatically, and it has become exponentially more difficult to run your own mail server." Okay. I don't know why that's the case. I run my own at GRC, and it's not hard at all.

Anyway, they say: "The complexity of running an email service is compounded by the antispam arms race that small mail operators are thrust into." Okay, again, not a problem here. Anyway: "At the very least, we'd like to lower the barriers to entry for running a functional, secure mail server." Okay. Yay for that. I'm all for it. I just don't know exactly what it is they're talking about. They say: "Beyond developing and testing Certbot plugins for popular MTAs, we're still brainstorming ideas for the decentralization of the email ecosystem." Ah, so I guess they don't know yet, either, but they have a goal. "If you work on easy-to-deploy MTA software," they say, "let's get in touch."

So then finally they finish with a call to arms. They say: "You can help, too." They write: "All of our software packages are currently in a developer beta state, and our team is stretched thin working on all of these projects. You can help make the email ecosystem more secure by preloading your email domain on our policy list, contributing to or reporting feature requests to STARTTLS Everywhere, helping implement and promote security features like DANE validation in MTA software, and contributing certificate installer plugins for MTAs to Certbot."

So anyway, that's STARTTLS Everywhere. Certainly it would be good to have it, and there's no reason not to, maybe as a fallback for, if nothing else, it provides opportunistic encryption which is probably supported by both endpoints over the existing ports for systems that are really old and creaky and know about STARTTLS, but don't for whatever reason want to connect to the new TLS ports, which are available across the board for these protocols. Anyway, a tip of the hat to the EFF. I'm glad, you know, they always have our backs, and I'm glad for the work they're doing. And that's the story on securing email.

Leo: What mail server do you use, if it's so easy?

Steve: I use hMailServer on Windows, which is very nice. It's free. Lots of features. My very, very, very favorite one is an anti-blacklisting feature. Any time, with zero tolerance, a server connects to mine and asks for a non-existent account, they are blacklisted.

Leo: Nice.

Steve: Because what the servers do is they guess. They just start at Abby and go down to Zeke. And so, as you know, Leo, I routinely obsolete my email addresses. And so what that does is any spammers are using an obsolete address from previous years, bang, they are blacklisted.

Leo: Nice.

Steve: So as a consequence, we get no spam. It really is a nice system.

Leo: You get a lot less mail, too, because nobody knows your address. But that's another matter entirely because you don't really care.

Steve: And that works for me. That works for me.

Leo: I would like to do the same thing. hMailServer.com.

Steve: Great.

Leo: Yeah. I've got to find something that simple for Linux. That would be nice. To run my own server would be nice.

Steve: Yeah.

Leo: And you don't need a lot of bandwidth to do that because you're not getting a lot of mail.

Steve: No, email is typically - yeah.

Leo: Can you reject HTML mail with it? You do, don't you?

Steve: Oh, yeah. You can do anything you want to.

Leo: You could do that, as a matter of fact, yeah. I think that's smart. My friends, we've come to the concluding portion of this show in which I remind you that you can watch us do it live every Tuesday, 1:30 Pacific, 4:30 Eastern, 20:30 UTC, at live.twit.tv or TWiT.tv/live. You choose. They're different players. Some people prefer one; some people prefer the other. In either case, though, you'll get access to a variety of different hosts. We use YouTube and Twitch, Ustream and Mixer. There's also audio streaming that we do ourselves. Or, no, maybe we use Spreaker. I think we use Spreaker and another service to do that now.

In any event, watch or listen live, if you can. If you're going to do that, be in the chatroom, irc.twit.tv. You can join the back channel. You can also be in studio. We have two great studio audience members who've been here all day. They're very patient. Email tickets@twit.tv if you're going to be in the Petaluma area, Petaluma, Northern California area, and come on by. We'd love to have you.

If you can't do any of those, you can always get a copy of the show from Steve's server, GRC.com. He's got audio and really nice transcripts. You can read along. And while you're there, check out SpinRite, the world's best hard drive, maintenance, and recovery utility. Free upgrades; right? If you buy SpinRite now, you're good through 7. Is that right?

Steve: Yes, sir. Yes, sir.

Leo: Don't want to put any words in your mouth. Okay.

Steve: Well, wait, wait. Through the 6.

Leo: Through 6. Seven will be an upgrade. Okay.

Steve: So you get all of the speed and compatibility improvements that are planned for SpinRite for the .x releases, yes.

Leo: Seven's a total new - yeah, yeah, yeah.

Steve: And actually 7 is going to blow the doors off of data recovery, but we'll save that [crosstalk].

Leo: Oh, I like that, hmm.

Steve: Yep, it's going to do everything that everybody wants in data recovery.

Leo: Oh, that would be very nice. Well, we'll find out more about that as time goes by. There's also stuff there that's free, that's fun, that's useful, that's informative. It's a great site. It's one of those sites that it's just you start, and you never - you continue to browse and browse and browse. There's always something new and interesting there. GRC.com.

You can reach Steve on Twitter. That's probably the easiest way to message him or tweet him. GRC is his company, Gibson Research Corporation. His Twitter handle is his initials, SG, plus GRC, so @SGgrc. That's the easiest way to reach him, but you can also go to GRC.com/feedback. We have audio and video. God knows why you'd want to watch, but you can. You know, watch every once in a while, and then you can get an idea of what Steve's moustache looks like today and things like that.

Steve: That's right. The important things.

Leo: The important stuff. Now, that's at TWiT.tv/sn. And of course, if you have a favorite podcast application, and they're all over the place, use that and just look for Security Now!, subscribe, and that way you don't even think about it. It'll just be there every Tuesday evening when you need it. Steve, thanks, have a great week.

Steve: Thank you, my friend. Talk to you next week. Bye.