The Mega FaxSploit

Description: This week we cover lots of discoveries revealed during last week's Black Hat 2018 and DEF CON 26 Las Vegas security conferences, among them 47 vulnerabilities across 25 Android smartphones, Android "Disk-in-the-Middle" attacks, Google tracking when asked not to, more Brazilian D-Link router hijack hijinks, a backdoor found in VIA C3 processors, a trusted-client attack on WhatsApp, a macOS zero-day, a tasty new feature for Win10 Enterprise, a new Signal-based secure email service, Facebook's Fizz TLS v1.3 library, another Let's Encrypt milestone, and then "FaxSploit," the most significant nightmare in recent history - FAR worse, I think, than any of the theoretical Spectre and Meltdown attacks.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-676.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-676-lq.mp3

SHOW TEASE: It's time for Security Now!, and it is our annual post-Black Hat/DEF CON episode. That means there are exploits galore. Macintosh, Android, and, yes, even your Hewlett-Packard printer. Steve will cut through the chaff, tell you what exploits to worry about, what ones aren't so bad, and give you all the security news, too, next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 676, recorded Tuesday, August 14th, 2018: The Mega FaxSploit.

It's time for Security Now!, the show where we cover your security, privacy, safety, and well-being - that's the new word everybody's using, "well-being" - online. Here's the Master of Well-Being, or MWB, Mr. Steven Gibson. Hello, Steve.

Steve Gibson: Yes, it used to be things needed to be user-friendly.

Leo: Not anymore.

Steve: Now we want well-being.

Leo: Yes. That's what Google calls the things that make your phone less attractive in the middle of the night, well-being.

Steve: Really?

Leo: Yeah. Digital well-being. And Apple and Google both are kind of all over that. Because you know why, because people don't sleep as much as they used to because they're Snapchatting all night, or something.

Steve: Huh.

Leo: It's not a problem you have, I'm sure.

Steve: No, it's not. So Leo. We have arrived.

Leo: What? We're here?

Steve: We are here. And this is not the penultimate, because I learned my lesson about that word. This is the ultimate episode of our 13th year.

Leo: Oh, you scared me. I thought we were at 999.

Steve: No, no.

Leo: Oh, we are completing our 13th year.

Steve: Yes. The first podcast we did was 18 minutes long, in 2005, on August 19th. And next Tuesday will be August 20th. So we will lap ourselves by one day, meaning that that's the beginning of Year 14, and this is the last podcast, no tears to shed, of Year 13 of Security Now!.

Leo: It did feel like an unlucky year.

Steve: It was, yeah, that's a very good point. And I have to say, too, that, I mean, it's been a rough year. And I know that you sort of laughed at this because you don't think people - I guess you don't assume that these fax combo scanner devices are in heavy use, or that people don't have fax machines.

Leo: Oh, I saw this story about people getting hacked on their HP printers through the fax. And I thought, well, who the heck has a fax machine hooked up to the phone?

Steve: So they're still very popular. I mean, like I know of five or six businesses. I have two myself. And 100% of businesses in Japan; 45% of households in Japan. Google lists 300 million fax numbers.

Leo: Well, you know why it's big in Japan. First of all, they invented the fax because they don't have Roman characters. They have to handwrite their characters. So it just makes sense to send faxes of handwritten notes.

Steve: Do you remember the Qwip? That was the name of that first machine? It was the Qwip.

Leo: Yes, I do.

Steve: And it was a cylinder that you wrap - it, like, grabbed the paper along its long edge.

Leo: Oh, there's a blast, yeah.

Steve: And it spun. And then this head very slowly moved, sort of like - anyone who's used a fishing reel where the guide moves back and forth, well, this was the head very slowly moving down. And this image, it was amazing. You could sit there and watch it, and it would be spinning, and very slowly. I think it would have to have been a 300-baud modem because that's the technology we had at the time. And you got, after about, I don't know, 10 minutes or so, I mean, it was not fast, kind of an image.

Leo: This is how newspapers got pictures from the wire services was one of these, yeah.

Steve: Yes. So we are the week after the, I mean, well, I'm at a loss for words. This year's Black Hat 2018 and DEF CON 26 Las Vegas security conferences occurred last week. Cornucopia, there we go, that's the word I was looking for. It's just, whoa. We've got 47 vulnerabilities across 25 Android smartphones, an Android "disk-in-the-middle" attack, Google tracking when asked not to, more Brazilian D-Link router hijack hijinks, a backdoor found in the VIA C3 processors, a trusted client attack on WhatsApp, a macOS zero day, a tasty new promised feature for Windows 10 Enterprise, a new Signal-based, Signal protocol-based secure email service, Facebook's release of their Fizz TLS v1.3 library into the open source community, and another Let's Encrypt milestone met.

And then we're going to talk about FaxSploit, which I think is the most significant nightmare in recent history - far worse, I would argue, than any of the theoretical Spectre and Meltdown attacks that we have talked about because of the ability for it to target specific organizations with a very potent attack over the phone.

Leo: Wow.

Steve: So a good podcast.

Leo: Over the phone.

Steve: Over the phone.

Leo: Who would have thunk it?

Steve: Huh, what? Yes.

Leo: Do you want to do your Picture of the Week? I don't want you to forget now. Sometimes you forget, you know.

Steve: No, I'm not going to forget. I know. You're exactly right. So this is a screenshot I took myself this morning when I did this because I have a combo scanner/fax/printer, a black-and-white one, and I also have a separate HP color printer that I sort of use depending upon what I'm printing. And it was vulnerable, meaning that, had somebody known my personal fax number, and this will be what we cover in detail at the end of this podcast, it is possible for all HP devices, and probably other manufacturers, too. But the attackers, the hackers who developed this only reverse engineered an HP printer. It is possible to send a specially formatted fax which will take over the printer, install the NSA exploits, and to then take over the network to which that device is attached.

So if our listeners do nothing else - and I understand, maybe people have moved beyond fax altogether. Maybe they've moved beyond a physical fax device. I know that HP is selling them, and there's a bazillion models of them. The point is it strikes me as a perfectly targetable attack. Businesses have fax numbers like on their web pages. And if they have an un-updated HP device answering that phone, they're vulnerable unless the device has been updated or has updated itself.

Interesting, after I posted the show notes, I got a tweet from Joe Levine who said: "Interestingly, my printer, an Office Jet Pro 6970, would not update from the downloaded software. It turns out it had auto-updated itself sometime back and is now maximally up to date without my doing anything at all." So it may be the case that HP devices, as all such unattended things should, have already taken care of themselves. But you're going to want to make sure.

So the good news is, when I first encountered this I'm thinking, oh, how does one update some random device's firmware? It could not be simpler. And I've got the links in the notes. And in fact I also tweeted the link earlier this morning, along with an urgent request for people to consider this. You download an executable. You run the EXE. It finds the device on your network. You say yes, and it updates it. So you don't have to - there's not a log of rigamarole associated with it. I know of a number of businesses I work with who also have fax machines, and this is the first time all year, for example, or for years, and in some cases ever, that I have written them an email saying "Please forward this to your IT department."

If it affects you, it's serious. Because this is so - first of all, hackers are not going to be able to, like, not have fun with this. All the details were presented during DEF CON last week, necessary for bad guys to retrace these guys' steps. It was responsibly disclosed. HP has dealt with it. But we know, unless the printers are taking care of this themselves, and clearly some do, the devices are going to remain vulnerable. So if nothing else, if you're affected by this, take it seriously.

We will come back around, and I'll talk about it in detail at the end of the podcast. But this is the dialogue box, our Picture of the Week, that I saw this morning when my system updated. So I was happy to have that done. And it turns out my office device had been - it was an HP that got kind of flaky after many, many years of service. So we switched to a non-HP device, and this morning that was removed from our network. It

actually hadn't been used. Sue was still using the HP for the printer function, I think. But anyway, it's now off of that network because at this point, as far as I know, all non-HP devices are vulnerable, as well. And I'll talk about more of that.

Leo: Interesting. So if you have a printer with a phone line plugged in, and you don't use it for faxing, unplug the phone line. That would eliminate this problem.

Steve: Yes. Yes, yes, yes.

Leo: And then just plug it in when you need to fax.

Steve: Yes. And of course this is, again, this is our constant refrain. It is a massive interpreter in the fax machine which interprets the fax command system. And as we know, interpreters are notoriously problematic. And this is a very, very cool hack, so we'll talk about that at the end.

So in the meantime, during last week's Las Vegas DEF CON security conference, researchers with the U.S. mobile and IoT security firm Kryptowire, spelled with a "K," revealed findings from their research which was conducted as part of a grant awarded by our U.S. Department of Homeland Security. They presented the details of 47 vulnerabilities in the firmware and the default installed apps of 25 different Android smartphone models, 11 of which have a presence in the U.S. market. And this is one of those distressing cases where the vulnerabilities are literally too numerous to tick off one by one. So I've included a link to Kryptowire's announcement in the show notes for anyone who's interested.

Anyway, the vulnerabilities discovered on the devices, which were offered by major U.S. carriers, include arbitrary command execution as the system, that is, as root, obtaining the modem logs and Logcat logs, which is a feature in Android, this Logcat, wiping all user data from the device, which is essentially performing a factory reset, reading and modifying a user's text messages, sending arbitrary text messages, getting the phone numbers of the user's contacts, and more. And of course all of these fall outside of the formal Android permission model. The major brands affected were Alcatel, Asus, LG, Nokia, Sony, and ZTE. And then there are lesser brands, some I've never heard of, but Coolpad, Doogee, Essential, Leagoo, MXQ, Oppo, Orbic, Plum, SKY, and Vivo.

And so rather than getting into the details - oh, you'll notice that Google is not present in the list. And looking at this range of vulnerabilities, many in the firmware, which we know means they will probably not be fixed, especially by lesser brands that are sort of off the map and are happy to take your money, and then you never hear from them again, I was sort of asking myself what's best? Is it better to have a monoculture or a heterogeneous spread of devices? We know that the danger of a monoculture approach is that, when there's a problem, it affects everything. Well, like HP; okay? Clearly, HP had one core of code to handle their fax interpretation, and it spread. You know, they used it in all of their devices. And they're a major longstanding provider of that technology.

So this one bit of research suddenly affects all HP combo fax/scanner/printer devices everywhere. Which you could argue, whoops, I mean, that's a problem. And we've seen various, you know, like Heartbleed sort of had this problem because it was a problem in the most common web server family that shared a common code base. So there's a danger with the monoculture approach that you don't have with heterogeneous spreads. For example, of course it's incredibly impractical, but if everyone wrote their own operating system, like individually, then there wouldn't be - first of all, then they would

all be insecure. But they'd all be insecure in different ways. So you wouldn't have the concentration effect of finding a vulnerability, for example, in all of the D-Link routers in Brazil, which allows them all to be commandeered, that we'll be talking about later.

So I guess the question sort of is what's your individual profile? That is, are you likely ever to be targeted or not? The reason I think that comes up is that, if you were somebody, a political dissident or somebody who for whatever reason anyone might be focusing on, then the danger of using an off-brand device, and unfortunately by "off" I mean other than Apple or Google, is that the entity targeting you could probably figure out what device you have and then, as a consequence of this rich background of attacks, figure out how to get into that particular device in order to compromise you in a targeted attack. That sort of person, I would argue, is safest and probably only truly safe against that kind of focused target attack if you stay with the major provider for the platform.

Even Google is not safe. We're going to be covering a little bit later a story in which Google's own devices or apps on Android have some problems. They respond immediately and fix them; so that's, you know, as we know, from a security standpoint, that's the best we can really expect. Anybody can make a mistake. HP made a big mistake. They immediately fixed it. This was all disclosed with HP responsibly, and patches have been out for weeks now, before this went public. So again, we're familiar with that cycle.

But so when somebody is deciding whether it's worth the money to purchase a more expensive smartphone, which is what you pay if you buy a Google brand device or an Apple brand device, I think the question is who are you because you really want, I mean, this is, as we know, these are pocket computers. And they are imperfect. And problems are being found. But what you want is, to the degree you need to be truly concerned about security, I just think staying with the platform's parent where there is a proven fast response to security problems is the only way to be safe.

Leo: I only use Pixel 2 phones because of that. I mean, LG is a major brand. Sony is a major brand. Essential is a major brand. These are all supposedly Android OEMs who do regular updates, who work hard at security. And I always say never buy an Android phone if the manufacturer hasn't agreed to push out the monthly security updates that Google offers in a [crosstalk] fashion. But honestly, these guys, some of these guys are on that list. So I would say at this point you're right, get a - maybe Samsung. I notice Samsung's not on there.

Steve: No. Very good point, yes.

Leo: But I would say Google. I mean, and the goodness is that Google Pixel phones are as good as anything else out there; the best cameras out there. So wow. Yeah, I pretty much stick with the Google phones, and this is a good reason.

Steve: Yeah. So here's another reason. Check Point Research, we're going to be hearing their name a lot in the next hour and a half. They have been really busy. In this case, they examined an interesting shortcoming in the way Android apps used storage resources. It turns out that sort of the careless use of the external storage by applications on Android can open the door to attacks resulting in any number of undesired outcomes: silent installation of unrequested, potentially malicious apps to the user's phone; denial of service for legitimate apps; and even causing applications to crash, which can open the door to, as we know, the beginning of an exploit as it crashes. As it matures, it no longer crashes, it runs the attackers' code for them.

So the problem is the nature of the way Android is currently handling the storage resources on its devices. There are two types of storage, loosely: internal, with each application using its own internal space which is segregated and enforced by the Android sandbox; and then there's external storage, which is often an SD card. I know on my Android devices, I mean, I've used it this way. It's an SD card stuck in that can be big, and it's exchangeable and so forth. And it creates a logical partition within the Android OS. And that region, and here's the crux of the problem, is shared by all applications. That external storage is often used to deliberately share files between apps or, for example, with a PC. So it is a sandbox bypass by design. It gives you some flexibility.

What Check Point researchers asked and then answered is how can we use this? Can we use that to effect exploitation of the Android trust, essentially, the trust boundaries? And in their coverage they explained that there are other reasons for an app developer to choose to use external storage, rather than the sandboxed internal store. It might be lack of sufficient capacity in the internal storage. Maybe backward compatibility considerations with older devices. Or not wanting the app to appear to use too much space inside the phone. Or just lack of care on the developer's part.

Anyway, so whatever the reasons, when using the external storage, certain precautions are necessary. And Google does understand this, even if they don't yet enforce it. According to Google's Android documentation, application developers are advised about their use of external storage in their apps with guidelines which include "perform input validation when handling data from external storage; do not store executables or class files on external storage; and external storage files should be signed and cryptographically verified prior to dynamic loading." Right? I mean, all those things make sense.

Well, it turns out that what Check Point has called "man-in-the-disk" attacks are made possible when applications are careless about their use of this Android external storage, such that failing to employ pretty much any of those security precautions can leave applications vulnerable to the risks of deliberate data manipulation. And apps from major OEMs, including Google, do not always follow these guidelines. During their research, they found instances where an app was downloaded, updated, or received data from the app provider's server, which was passed through the external storage before being sent on to the app itself.

So the app used the external storage as a temporary staging store for a blob of update code, or app modification code, before incorporating it into the internal store. And of course doing that violates Google's guidelines and offers, I was going to say provides, an opportunity for an adversary to manipulate the data held in that store during the interval before the app reads it again. So they actually developed exploits for all these things. Applications that were tested for this new attack surface included Google Translate, Yandex's Translate, Google Voice Typing, Google's Text-to-Speech, the Xiaomi browser, and various other apps.

After referring to the advice given within Google's guidelines, Check Point compared the advice to what was actually the case. The Xiaomi browser was found to be using external storage as a staging resource for application updates. Check Point was able to execute a successful attack by which the application's update code was replaced, resulting in the installation of an alternative undesired application instead of a legitimate update.

So literally in this case they downloaded an app that looked innocent. I think it was a flashlight app. And the flashlight app asked for access to external storage. Since that's not regarded as a security issue or problem, and apparently lots of Android apps that you download and install ask for that permission, it's often given. So that's all an app needs in order to exploit this kind of attack is, no matter what kind of app it is, access to external storage. It then monitors the external store, waits for an opportunity, replaces

the staged download, and its modification of what was downloaded gets installed instead, and nobody is the wiser.

In the case of Google Translate, Yandex Translate, and Google Voice Typing, Check Point found that the developers were failing to validate the integrity of data read from external storage. So those apps were also using external storage, but assuming it was pristine, that is, what they had loaded there hadn't been changed. So they were taking it as gospel when they brought it in and did whatever they do with it, interpreted it in some fashion. So that Check Point was able to compromise certain files required by those apps and get them to crash, which as we know could lead to eventual takeover.

After the discovery and verification of these problems, Check Point did responsibly contact Google, Xiaomi, and vendors of other vulnerable apps to update them and request their response. Google immediately responded and fixed the problems. Xiaomi chose not to address it at all. And as I understand it, the Xiaomi browser is a very popular browser on Android, which is now known to be vulnerable to this sort of an external storage staging attack. And apparently it's going to remain so.

So to conclude, Check Point summarized the problems and what they view as shortcomings of Android by saying: "An Android device's external storage is a public area which can be observed or modified by any other application on the same device. Android does not provide built-in protections for the data held in the external storage. It only offers developers guidelines on proper use of this resource." But again, no enforcement.

"Developers anywhere," as we know, "are not always versed in the need for security and the potential risks, nor do they always follow these guidelines. Some of the pre-installed and popularly used apps ignore these guidelines and hold sensitive data in the unprotected external storage." And again, that is subject to manipulation. And they say: "This can lead to a man-in-the-disk attack, resulting in the manipulation and/or abuse of unprotected sensitive data." And, as they found and proved: "Modification to the data can lead to," as they put it, "unwelcome results on the user's device."

So this probably doesn't come as a huge surprise, but it's a warning. And I would argue that Google needs to go further than just tell developers, you know, be careful about what you read from external storage. Don't assume it hasn't been modified. Certainly anybody, I mean, the only way, in the case of the Xiaomi browser, could be caught out is if it is not verifying a signature on the thing it downloaded and then is incorporating into itself from external storage. I mean, that's unconscionable, but it is happening. And who knows what other apps that may be happening to. It's very simple for somebody to install an app that says, oh, I'm going to need to have access to external storage; okay? And most users are going to say yes.

So hopefully, maybe what we need is for Google to extend the sandboxing of external storage out into it so that there is per app ties. And then where apps explicitly need to share through external storage, that permission is granted because either that or they have to somehow - I don't think an app can be forced to parse the data it reads. Maybe it could be forced to have executable content signed.

But again, I mean, we have a huge ecosystem of Android apps, and this is a glaring vulnerability which I'm really happy that Check Point brought to light. Google is aware of it. Maybe they'll respond. But until then, I guess all I can do is caution our listeners who are Android users that this is a potential vulnerability that's just hanging out there right now.

And while we're on the topic of Google and Android, it turns out that the Hacker News reported on some research that the AP, the Associated Press did, which is probably, I mean, this is maybe a little inflammatory clickbait, but it's worth pointing out to our

listeners. The statement was "Google tracks you everywhere, even if you explicitly tell it not to." And there is some truth to it. What the AP found is that preventing Google from acquiring long-term tracking data requires more than turning off the obvious location history function. What the Hacker News wrote was: "Every time a service like Google Maps wants to use your location, Google asks your permission to allow access to your location if you want to use it for navigating." They say: "But a new investigation shows that the company does track you anyway."

The AP revealed that many Google services on Android and iPhone devices store records of your location data, even when you have paused the location history on your mobile device. Disabling location history in the privacy settings of Google applications should prevent, it is argued, Google from keeping track of your every movement as its own support page states, quote: "You can turn off location history at any time. With location history off, the places you go are no longer stored." That's not true.

As the AP explains, quote from the AP: "For example, Google stores a snapshot of where you are when you merely open its Maps app. Automatic daily weather updates on Android phones pinpoint roughly where you are. And some searches that have nothing to do with location, like 'chocolate chip cookies' and 'kids science kits,' pinpoint your precise latitude and longitude," they write, "accurate to the square foot, and save it to your Google account."

During their coverage, to demonstrate the threat of this, what they consider - I mean, again, clickbait, but worth understanding - of this Google practice, the AP created a visual map of the movements of a Princeton postdoctoral researcher, a Dr. Acar, who carried an Android smartphone with location history switched off to prevent location data collection. However, the researcher discovered that the map includes records of Dr. Acar's train commute on two trips to New York and visits to High Line Park, Chelsea Market, Hell's Kitchen, Central Park, and Harlem. To protect the privacy of Dr. Acar, the publication did not plot the most telling and frequent marker on the map, his home address. According to the researchers, this privacy issue affects around two billion Android users and hundreds of millions of iPhone users across the world who rely on Google Maps or search.

In response to the AP's investigation, Google issued the following statement: "There are a number of different ways that Google may use location to improve people's experience, including location history, web, and app activity, and through device-level location services." They say: "We provide clear descriptions of these tools and robust controls so people can turn them on or off and delete their histories at any time."

Jonathan Mayer, whom we have spoken of years past often, a Princeton researcher and former chief technologist for the FCC's enforcement bureau, argued: "If you're going to allow users to turn off something called 'location history,' then all the places where you maintain location history should be turned off." He says: "That seems like a pretty straightforward position to have."

So for what it's worth, to stop Google from saving timestamped location markers, users need to turn off another setting, and that's that web and app activity, which is enabled by default and stores a variety of information from Google apps and sites to your Google account. Once disabled, it will not only stop Google from storing location markers, but also prevents the company from storing information generated by searches and other activities. So anyway, I just wanted to point this out. It seemed a little over the top in the way it was being described. But it's certainly reasonable, if somebody wasn't really paying attention and thought that just turning off location history shut that down completely, you also need to go into your Google account and turn off web and app activity, which does not just record activity, but where that activity occurred.

So, okay. We've been talking about D-Link routers in Brazil. Let's see. Last time we talked about them, they were being commandeered into a botnet in short order, within 24 hours. I coined the term, although it wasn't particular, didn't require much imagination, a flash botnet because it was just flashed into existence within a day. As we know, these D-Link routers have - many of them, unfortunately, tens of thousands - have a known old vulnerability which allows remote access to essentially an authentication bypass that gives somebody on the outside access to running their own code and commandeering the router remotely.

Well, Radware's Threat Research Center has just identified another hijacking campaign also aimed at Brazilian D-Link routers against two Brazilian banks. This is an attack which hijacks - these are DSL modems, D-Link DSL modems. It hijacks their DNS settings. And we've talked about DNS hijacking in the past. In this case the DNS servers on these vulnerable routers are pointed to a malicious DNS server which replaces the proper ID for two domains: www.bb.com.br, which is Banco de Brasil; and another one is Itau Unibanco, and that's www.itau.com.br. In both cases, the IP address redirects to a fake website which then attempts to spoof the user and to obtain their bank credentials.

Now, okay. We know there are limitations to this sort of attack; right? We know that, unless the attackers are able somehow to obtain valid TLS certs for those domains, it won't be possible for users to avoid seeing a warning of some sort, and in often cases just like hard to bypass warning. I mean, the good news is - we were complaining last week about how difficult it is to bypass these warnings when you legitimately know you want to, like you're wanting to talk to your own LAN router over HTTP, and your browser is becoming increasingly resistant to allowing you to do that. That's a pain. In this case, you don't want to be fooled into making an HTTP contact to your bank.

So the problem is that, if banks also allowed HTTP and didn't force a redirect - well, in fact, even if they did force a redirect, if they allow an HTTP connection, which then bounces you over to a secure connection, it might be that people's bookmarks for their banks would be HTTP. That would allow these bad guys to establish a fake bank website which does not redirect users to HTTPS and of course doesn't have any other HTTPS links. If people are not paying attention, they wouldn't notice the difference and would interact with this cloned site without knowing that it wasn't where they thought they were and could get themselves in trouble.

We've talked about, in fact you offered it up last week, Leo, HSTS, HTTP Strict Transport Security. What should be done is that all companies that are intending only to be accessed over HTTPS add the HSTS header to every response from their server. That allows browsers to learn and permanently or semi-permanently, because you can set an expiration, but essentially the idea is forever, and it's refreshed every time you revisit, to learn that the bank only wants HTTPS connections such that the browser is given permission to automatically upgrade HTTP connections to HTTPS.

So the point is, you know, we're sort of in this crossroads period. We've got Chrome raising the alarm increasingly when any site, even a benign blog site, isn't HTTPS. We have all the mechanisms in place, but not yet completely deployed. Unfortunately, we have browsers that are not yet also trying HTTPS when they try HTTP. And I would argue it's about time for them to be making parallel lookups and connections to see if they can autonomously move over to HTTPS. There are some problems with doing that. So I can understand the need for the permission from the site in the form of an HSTS header. And GRC is on Google's and many other permanent HSTS lists. We got on the list very early when we first talked about this years ago, and so we will always be accessible over HTTPS.

So it's worth reminding people that this is a problem, that is, that the thing that can be done to avoid this is not, if you are worried that your router could be compromised, is

not to use DHCP, the Dynamic Host Configuration Protocol, which is the obtain IP address automatically. You can typically override your computer's DNS settings so that it doesn't get your DNS servers automatically from the router. You provide them yourself, in which case you would be immune from this sort of takeover. Certainly better to take responsibility for your router not being updated.

But maybe someone has no opportunity to do that. There is no updated firmware. The router's no longer being maintained. They're worried that there could be some exploit. As we know, even closing down WAN access, WAN management on that side doesn't provide bulletproof protection. So I could see, sort of as a belt-and-suspenders approach, just change your own machine's or any important machine's DNS over to something that you explicitly and deliberately specify so that, if your router had its DNS changed, you'd still be safe. But better to have a good router. And they're cheap these days, if you have any choice.

But notice that things like multiple NAT servers, network segregation, other things we've talked about to protect users in the past wouldn't protect you from this. Making sure, however, that you have an HTTPS connection to your bank and that you verify that's where you are would go a long way to keep you safe.

Also at DEF CON - it's pretty much a DEF CON podcast we have.

Leo: Yeah, it usually is after DEF CON.

Steve: Exactly, once a year, right around our anniversary. A repeat DEF CON presenter, Christopher Domas, who is an annual contributor, I mean, he's a hardware hacker, specializes in x86 stuff. His presentation this year was titled: "God Mode Unlocked: Hardware Backdoors in x86 CPUs." He revealed the presence of a hidden, as he called it, "god mode," which was undocumented in the VIA 3C x86-based CPUs, at least back from 2001 through 2003. There have been people that have argued, well, it's not really a backdoor because they disclosed it in 2004. This feature was present in the VIA C3, it looks like, what, Nehemiah, N-E-H-E-M-I-A-H, chips.

But Chris said that all other C3 chipsets are bound to feature a similar mechanism. He actually named this god mode the "Rosenbridge backdoor," which allows an attacker to elevate the execution level of malicious code from ring 3, where security safeguards would constrain it to kernel ring 0, where it's unrestrained. And if Rosenbridge sounds familiar to any of our science-oriented listeners, that's because the official name for a space-time wormhole is an Einstein-Rosen bridge.

Leo: Aha.

Steve: Which suggests the possibility of short-circuiting space time if a nontrivial solution to Einstein's field equations can be shown to hold. So Einstein presented the possibility of such a thing, although we sci-fi buffs, I mean, it's a constant, you know, for example, our favorite author, Peter Hamilton, his whole Commonwealth...

Leo: Is based on that, isn't it, yeah, yeah.

Steve: ...is knit together very cleverly with wormholes which short-circuit space time. Anyway, Domas says that this backdoor mechanism is a RISC, R-I-S-C, Reduced

Instruction Set Computer, coprocessor that sits alongside the main C3 processor and is part of it. By using a launch instruction, which is a 0f3f, a register control bit can be flipped which enables this additional coprocessor. And Chris says that coprocessor does not enforce the same security protections as the main C3 chipset. And nobody disputes that.

This alternate instruction set says the documentation, which was subsequently published, this alternate instruction set includes an extended set of integer, MMX, floating point, and 3DNow! instructions, along with additional registers and some more powerful instruction forms over the x86 instruction architecture. So it's sort of a switch you can flip on which extends the chip's instruction set with formally defined instructions and then a bunch of apparently VIA's own instructions.

However, the VIA document also mentions that the additional instruction set is specifically meant for testing, debugging, or other special conditions. Hence the reason it is not documented for general usage. Which seems curious to me because certainly MMX, floating point, and the 3DNow! instructions, those are all general use instructions. They're not debugging and testing instructions. But it is nevertheless too potent to be left available because any instructions sent to this additional coprocessor is run under ring 0, bypassing all security and privilege barriers, rather than under the normal ring 3.

The good news is that this controversial backdoor, as Chris himself explains, should require kernel-level access to activate. So there was the intention of some protection. However, he points out that this Rosenbridge backdoor mechanism, as he named it, has been observed to be enabled by default on some systems, allowing any unprivileged code to modify the kernel without prior exploitation. In these scenarios, the attacker only needs to send that specially crafted instruction to the additional RISC processor which was then standing by, ready to read and execute them with no ring 3 privilege restriction.

So I'm glad he brought this to life. The problem is these chips, I mean, they are - it's a very popular sort of low-end Intel x86 clone present in all kinds of stuff. So if it happened to be on by default, this is a potentially serious execution opportunity. For example, it might well be that some of the printers and fax machines use these. Actually, the HP is a 32-bit ARM processor. We know from the reverse engineering research. But still, these little VIA C3s have been long used in lower end applications. I've got a link for additional information in the show notes if anyone's interested. Chris put out, I think it's a 110-page slides PDF at DEF CON.

Okay. So there was a bunch of coverage of an attack on WhatsApp. And I thought, uh-oh, that's not good. WhatsApp, as we know, was purchased by Facebook. It's arguably now, as a consequence of that purchase, what is it, the number one messaging app in use? And so the idea of somebody being able to spoof users and spoof what they send and inject malicious content into the conversation seems like a problem. Turns out, eh, useful research, interesting research, nothing to worry about. And the WhatsApp people say, yeah, we know, and we're not changing it.

Once again, Check Point researchers, who as I have mentioned have apparently been working overtime recently, revealed that someone who is already participating in a two or more way WhatsApp chat can alter messages to spoof their content and sender. Okay. Once again, somebody who's already trusted and has the keys required to chat in a two or more way messaging group can spoof the content and sender. So, okay. The press coverage suggests that this is a flaw in WhatsApp which takes advantage of a loophole in WhatsApp security protocols to change the content of the messages, allowing malicious users to create and spread misinformation or fake news from what appear to be trusted sources. Certainly I agree that that's the case. It turns out it's by design. It's not something that was overlooked.

Essentially what Check Point did was to reverse engineer some undocumented protocol. After being trusted, that gave them the access that they wanted. So by decrypting the protocol received by somebody who is in the group, the Check Point researchers were able to develop three attacks. And I've put "attacks" in quotes in the show notes because it's like, okay, first, they can change a correspondent's reply to essentially put words in their mouth, that is, change what they said; change the identity of a sender in a group chat, even if they're not a member, so like fake somebody breaking into the chat; and send a private message in a chat group, but when the recipient replies, the whole group sees it. So, that is, to unprivate a message in the chat group.

So basically they're screwing around with the internal messaging protocol in some ways which it probably should not surprise anyone is possible if you are participating in the group. So when the researchers reported the flaws to the WhatsApp security team, WhatsApp argued that, since these messages do not break the fundamental functionality of the end-to-end encryption, users always have the option of blocking a sender who tries to spoof messages, and they can report problematic content to WhatsApp. In other words, yeah, okay.

WhatsApp replied: "These are known design tradeoffs that have been previously raised in public, including by Signal in a 2014 blog post" - I think we covered it at the time, four years ago. And, they said: "We do not intend to make any change to WhatsApp as a result." So I guess WhatsApp users might be usefully cautioned against this possibility, that is, that it's possible for somebody in a WhatsApp messaging group with enough technical skills to mess with the content of that messaging of those people in the pool, in the group. So that, but nothing more.

Leo: The only reason it might be a cause for concern is that WhatsApp is a big culprit in the spread of phony news and disinformation in some countries like India.

Steve: Ah, okay.

Leo: I don't know how it works. But I would guess that WhatsApp, you'd be in a group, a large group of news. And that, in that case, you could be, for instance, somebody could post some misinformation and then fake a lot of other people in that group going, oh, you know, that's true. I saw that, you know, or that kind of thing.

Steve: Yeah, very good point.

Leo: So I wonder if that's what they're concerned about. I'm not sure. But that's a real problem in India and other places where WhatsApp has been used to incite mob killings and things.

Steve: Okay. I didn't realize that WhatsApp groups got that big. Because, you're right, if it's three people you know, then it's, yeah, eh. But if you're joining a massive messaging collective...

Leo: Well, that's the thing I don't know. I'm not sure how - I don't know if it's being - so this is a story from The New York Times from a couple of weeks ago, how WhatsApp leads mobs to murder in India. And these fake stories get spread. But I'm not sure if they're spread in messages. It says "Local authorities have struggled to

contain false messages which have circulated throughout India for months." So what I don't know is if it's a large newsgroup. If that were the case, though, that would be one reason that this would be something to be worried about; right?

Steve: Yeah, because, for example, a legitimate news provider could...

Leo: Be spoofed.

Steve: Could be spoofed completely, their identity and the content of what they're sending. So, yeah.

Leo: Yeah, that's something to pay attention to. I just - it says: "WhatsApp's design" - this is the Times again - "makes it easy to spread false information. Many messages" - oh, here it is - "are shared in groups. And when they are forwarded, there is no indication of their origin. The kidnap warnings have often appeared to come from family and friends." So it's the spoofing of messages that is part of the problem here.

Steve: Yeah.

Leo: Anyway, I don't - just more information on that story, I guess.

Steve: So I heard you on the tail end of MacBreak Weekly talking about last week's security conference relative to macOS hacks.

Leo: Yeah.

Steve: And I don't know whether one of the things you talked about was this synthetic mouse clicks problem?

Leo: Yes, we did, yeah.

Steve: Okay. So High Sierra is vulnerable, but Mojave will not be. Patrick Wardle, who is an ex-NSA hacker who is now the Chief Research Officer of Digita Security, discovered by accident and revealed a - and actually it's a zero-day, technically, because it came as a surprise to Apple. He did not responsibly disclose this.

Leo: He didn't tell them, yeah, yeah.

Steve: He revealed a critical zero-day vulnerability - which, by the way, is zero-day because of him - in the High Sierra macOS operating system that could allow a malicious application installed in the targeted system to essentially virtually click objects without any user interaction or consent. Okay, now, the reason that's a problem, I mean, so Mac has this technology built in for their accessibility features to help disabled people interact

with the system interface. But they explicitly hold security-critical events out of that. That is, for example, you can't use a synthetic mouse click to click on the "Okay" for a crucial security permission like access to the keychain or access to install something.

Normally events in an OS are typically discrete. And so, for example, there's a down click event that the mouse button makes when you press it down. Then there might be a series of mouse move events while you drag something somewhere. Then there would be a mouse up event when you release the mouse button. So these events are all discrete. What Patrick accidentally discovered was that in High Sierra, two consecutive synthetic, that is, non-physical mouse generated, but software generated, mouse down events in a row are treated as a legitimate click, which then allows attackers to programmatically interact with security warnings which ask users to choose between "Allow" or "Deny" access to sensitive data or features.

So I don't understand why Patrick chose not to disclose this responsibly. I mean, this is not good. The good news is Mojave, as a consequence of a redesign of this whole system, already has this mitigated by blocking all synthetic events. But for the time being, I mean, I imagine Apple will quickly move to address this. So in the meantime, I guess, be careful.

Leo: I've used this, and I wonder why, I mean, one of the reasons it's in there is for people with accessibility issues to be able to do stuff. And the way I've used it, this is - I didn't think this was a bug. But at one point, probably because of something I was using like text expander on my Mac, I wasn't able to click - you know when you install something very low level, a kernel extension on macOS, you have to then go into - this is really a nice, I think a nice feature.

It's not merely a warning, oh, are you sure you want to do this? You actually have to go into the system preferences. You have to go to security. It's where Gatekeeper lives. And you'll see a new thing that says, oh, this XYZ program is trying to install a kernel extension. Do you want to allow this? And there's a button that says "Allow." And you actually, I mean, this is a bit of a, and I think a good thing, a rigamarole...

Steve: Yes.

Leo: ...in order to allow something to modify your system. But for some reason, because of something I had installed, the ability to click on that "Allow" button was disabled. I clicked, click click, nothing happens, click click click. So I wrote an AppleScript. I figured out where the button is on the screen and wrote an AppleScript to click on that location on the screen. This is that exact facility we're talking about.

Steve: Yes, yes, yes.

Leo: And it's the only way I could click that "Allow." But if somebody wrote a malicious AppleScript to do that, you could see what the problem would be.

Steve: Yeah, to automate the whole process.

Leo: Right.

Steve: Of getting something into your kernel.

Leo: We talked about AppleScript and security on Friday on this Triangulation episode with Sal Soghoian, who was the king of AppleScript at Apple. And he even talked about the negotiation between the Automation Division, his division, and Apple Security, who's really good, about all this because it's really great from the point of view of automation to have access to Apple events and to be able to trigger events in a scriptable fashion. And so what they decided, apparently, was, well, the AppleScript would only run at the user level, the permissions level, the privilege level of the user running the script.

Steve: Right.

Leo: Which was somewhat mitigated because it prevented it from doing anything an administrator access was required for. It sounds like now they've even gone a step further. It's a tricky thing because you do want to give people automation capability, and you want to give people who have mobility issues...

Steve: A legitimate need.

Leo: Accessibility issues, the ability to do this, as well. That's what mouse keys is all about. So I understand the - it's always a balancing act, isn't it. It's a fascinating story.

Steve: Yeah, yeah. So our friend Lawrence Abrams, who founded Bleeping Computer, published on his site that Windows 10 Enterprise would apparently soon be getting a new InPrivate Desktop feature. This feature would allow administrators to run untrusted executables in a secure sandbox without fear that that executable might or could make any changes to the operating system or the system's files. Lawrence quotes the Windows 10 Insider Feedback Hub as saying - and I should mention it's since disappeared. It said: "InPrivate Desktop (Preview) provides admins a way to launch a throwaway sandbox for secure, one-time execution of untrusted software. This is basically," it writes, "an in-box, speedy VM [Virtual Machine] that is recycled when you close the app."

He writes: "The quoted quest is no longer available in the Feedback Hub. But according to its description, this feature is being targeted at Windows 10 Enterprise and requires at least 4GB of RAM, 5GB of free disk space, two CPU cores, and CPU virtualization enabled in the BIOS. It does not indicate if Hyper-V needs to be installed or not. But as the app requires admin privileges to install some features, it could be," he writes, "that Hyper-V will be enabled."

He wrote: "When the quest was live, I had attempted to install the InPrivate Desktop (Preview) app, but it was not accessible from the Microsoft Store as described. Furthermore," he says, "a wiki link in the quest description brought me to a page asking me to log into my Microsoft account. When I logged in with my account, I received a message that indicates I need to be part of the Azure Active Directory tenant for Microsoft." He says: "It's too bad that I was unable to test this feature as it looks to be an interesting way to execute untrusted software without fear of permanent file modification, program installation, or configuration changes. This will also provide a new

security boundary that Microsoft will need to protect and that researchers will be hammering on for bug bounties."

And I would add that it would be nice if this were to migrate down to other non-enterprise builds in the future, since I could see, I mean, for myself I'm often firing up a VM in order to run something whose behavior I want to constrain until I'm able to vet it a little bit. So that would be cool if that were built in.

I know this is going to be of interest to people, and I didn't know if it would hook you, Leo. But there is a new email service using the Signal protocol. It's called Criptext, or Cript, well, there's only one "T." Okay. So here it is. You probably want to bring this up: www.criptext.com. C-R-I-P-T-E-X-T dotcom. Criptext.com. And they say: "Criptext is an encrypted email service that guarantees security, privacy, and control over all your email communications. We don't have access to your emails, nor do we store them in our servers." They say: "You're in control now."

So under their bullet points they offer "End-to-end encryption: All your emails are locked with a unique key that's stored on your device alone, which means only you and your intended recipient can read the emails you send. Signal protocol: The Criptext email service utilizes the open source Signal protocol library, which protects your privacy and security throughout your entire Criptext experience. Open source: Criptext's source code is open to the entire privacy community to see. We actively work with our open source community to improve on the software in order to provide the best email experience. No cloud storage: Criptext doesn't store any emails in its servers. All your emails are stored on your device alone, which means you're in control of your data at all times."

Then they say: "We know that privacy isn't just about encryption, which is why Criptext doesn't store any emails in our servers." It's available on web and mobile: Mac, Windows, Linux, iOS, and Android. Tip of the hat to Bleeping Computer for pointing this out. And I just wanted to bring it to our users' attention. We know Signal protocol is arguably the best piece of work that exists. This was Moxie Marlinspike and his group put this together. I initially, our listeners will remember, thought that Signal protocol was a little overdesigned. Then I understood how it works. I'm not exactly sure how they say they don't store any emails on their servers. Email is essentially asynchronous, and it has traditionally been a store and forward. Maybe they just mean they don't permanently store it. So it's there in transit through them, kept encrypted until your recipient checks in or polls...

Leo: It's basically Signal. I mean...

Steve: It is. It is, yes.

Leo: It's just Signal. I mean, it's a messaging service. It's not an email service. It's Signal. Right?

Steve: Well, it's an offline Signal.

Leo: Yeah, but Signal's offline. If I'm not online, and somebody sends me a message...

Steve: That's true. In fact, that's why there's that whole collect a whole bunch of extra crypto keys for somebody who's offline that allows you to encrypt for them until they're online. So, yeah. I don't how it differs from Signal. Maybe it's just larger documents?

Leo: Yeah.

Steve: Maybe it's like more of an email UI than a short signaling protocol.

Leo: Is it web interface only? Or can you use an email client?

Steve: No. Web, well, good question. They say "web and mobile." But maybe they've implemented [crosstalk]. So I don't know...

Leo: But you can download something.

Steve: Yeah, then maybe they have an iOS and an Android app for it.

Leo: Right now it's Download for Mac is the only button. Oh, but you can side - open criptext.com/download on your phone to install. Apple doesn't allow sideloading, so I'm confused. Well, I'm going to download it on the Mac and see. But it sounds like it's just Signal, basically.

Steve: Well, it might be Signal oriented towards a larger format of transaction.

Leo: Yeah, exactly, yeah.

Steve: So what I would say is, given that they haven't made any mistakes, that is, given that it is Signal...

Leo: It is open source, anyway, so...

Steve: It is open source. Signal is as robust as it gets. It was really done well. If somebody wanted - and again, each end needs to have it. So you've got to have, in your own encrypted email a la Signal network, who you're sending it to needs to set themselves up, too. But this looks like a good way to send larger bulk stuff in a truly, as I said, as good as it can get sort of fashion.

Leo: Yeah.

Steve: C-R-I-P-T-E-X-T dotcom.

Leo: When I go to Criptext.com/download on my phone, it says 404.

Steve: Okay.

Leo: But I was able to download a DMG package for...

Steve: Oh, for the Mac.

Leo: ...Macintosh, yeah.

Steve: So there is some...

Leo: It looks like the only version right now.

Steve: Okay. It's very fresh.

Leo: Very fresh. Still fresh.

Steve: So in a very nice move, I think, Facebook has open-sourced their Fizz, F-I-Z-Z, TLS v1.3 Library. And I just give props to them. As we know, TLS 1.3 is the recently ratified latest version of TLS, which incorporates a bunch of new features for encrypting the handshake messages to keep certificates private, which otherwise aren't; redesigns the way secret keys are derived to make that more robust; offers a zero roundtrip connection setup which is able to make some requests faster than any previous technology. It was written in C++ 14. This is what Facebook is using. They say it's reliable and a highly performant TLS library that supports all major handshake modes, robust encryption algorithms, and performance optimizations which aim to transfer data securely over 10% higher speed than TLS 1.2.

They write: "With zero copy encryption and decryption" - meaning in place - "tight integration with other parts of our infrastructure, and other optimizations, we see a reduced usage of memory and CPU with Fizz. In addition to the enhancements that come with TLS 1.3, Fizz offers an improved solution for middlebox handshake failures, supports asynchronous I/O by default, and can handle scatter/gather I/O to eliminate the need for extra copies of data." And then they say: "Facebook has already replaced its older custom protocol, Zero, with Fizz, which is now responsible" - get this - "for securing trillions of connections every day at Facebook." In other words, I can't think of a stronger endorsement for something than it's what Facebook is already using, and it works.

The social media giant Facebook says it has deployed Fizz and TLS 1.3 globally in their mobile apps, Proxygen, their load balancers, their internal services, and even their QUIC, that's Q-U-I-C, the DNS-based library, which is MVFST. More than 50% of their Internet traffic is now secured with TLS v1.3. And it's on GitHub, GitHub.com/facebookincubator/fizz. So big props to Facebook for putting that out there. It's just very, very cool to have a robust and proven TLS v1.3 library now freely available for the Internet community to use.

And lastly, Let's Encrypt. I said that they had another milestone to report. They just reported that they are now officially trusted by all major root programs. Remember that when they started out they had their own root certificate, but nobody else knew about

them. I mean, it's only by having the root store or the root certificate in everybody's root store that we are then able to verify the signature of any certificates that the Let's Encrypt system signs. So to solve that problem, the root was cross-signed by InTrust. InTrust? I think it was InTrust. Oh, IdenTrust. So, and IdenTrust already had established a longstanding trust relationship across all platforms.

So as a consequence of the cross-signing, Let's Encrypt was trusted, not because they had signed it, but because IdenTrust had signed it. Now they report last week that they are in all the major root stores. So they will continue to be cross-signed for those devices which are not being kept current for whatever reason. But for all current devices and future devices, Let's Encrypt's root is in the store everywhere. And I imagine way downstream at some point, when their metrics demonstrate that nobody is any longer needing IdenTrust's cross-signature, they'll remove it. But another nice milestone for Let's Encrypt.

And speaking of nice milestones, this morning, or I guess it was yesterday morning, Monday, I found a nice tweet from Jim Berry that started off "Testimonial, so we know what that's going to be about. He says: "I have an HP Z200 used in a high-priority security context at my job that was experiencing the 'click of death.'" Okay, now, in this case this is a hard drive doing the click of death, and that's not something you...

Leo: There's a name from the past.

Steve: Yes, exactly. And that's not something you want. He says: "It was experiencing the click of death, and it wasn't booting at all." So this is an HP Z200 in a high-priority security context that was experiencing click of death and not booting. He says: "I was able to see the drive in Linux, but Windows was a no-go." He says: "I ran SpinRite on Level 4 over the weekend with the 500GB drive." And he says: "It still clicks," he says, "getting it replaced ASAP, but now it boots up. So the drive is being imaged onto a replacement, and the machine will be redeployed with the hearty thanks from my customer."

So that was interesting. So that's probably SpinRite bringing the drive back from just about as far over the line and the brink of death as is possible. What that suggests is that there is some serious, serious damage on the drive, probably caused by past power failures while the drive was writing. That's the kind of damage that can occur because what it suggests is that the servoing information, the low-level data on the drive, which we can't get at from the data level typically, is damaged. SpinRite nevertheless brought it back to the point where it was able to at least boot and get reimaged.

And you definitely - this here's one you don't want to say, oh, I wonder how much life left the drive has. Should I replace it after running SpinRite or not? No. You do a Hail Mary and reimage the drive. And the good news is SpinRite was able to get it to the point where it could be imaged and so the device experiences minimal downtime. So, yay, and thanks for sharing that, Jim.

Leo: Yay, yay, yay. All right. Steve, let's find out why we should disconnect the old fax printer.

Steve: So I do feel like I've pretty thoroughly already stepped on the lede here because - and frankly, I've done so only in the best interest of our listeners because I will be surprised if - I don't know even if this is going to get attention because it just might be used in targeted attacks. But it's been a long time since I've seen a vulnerability that has

such exploit potential for attack targeting. We often talk about how interpreters are so difficult to secure. It turns out that, once again, Check Point Research, as I said, working overtime, did some amazing reverse engineering of an HP combo fax/printer/scanner device.

The device was not easy to get into. I mean, not only physically. Physically they destroyed it. They had to cut off the plastic back in order to access the brain board inside of this combo device. They found the JTAG interface, which is the programming interface for connecting a debugger to an embedded processor. They were able to begin to see what was going on inside. But they needed to get more debugging capability in. So they found a means for bootstrapping themselves using a network-based large packet which they were then able to use to attack the printer from the network, all of this just for the research. This is just the reverse engineering part. This is not the attack. This is gaining a foothold for reverse engineering and understanding what was going on.

That allowed them, using first the JTAG to then figure out how to get a buffer overflow from the network, allowed them to get some debugging capability uploaded into the printer that was otherwise locked down. Once they were in there, they were able to more conveniently look around and start to understand what their environment was. They found an ARM 32-bit CPU which was running in Big Endian mode, meaning that the sequence of bytes is not least significant byte first, it's most significant byte first for multibyte values. They found that a shared memory region is used to communicate with the microcontroller and the LCD.

The operating system was ThreadX based, which is a real-time OS made by Green Hills. It uses a flat memory model, where there are many tasks running in kernel mode, all sharing the same flat virtual address space. And they determined that there was no ASLR mechanism. It was all fixed-position memory, which is probably reasonable for a turnkey device like that. They then proceeded to look for vulnerabilities from the phone line. I mean, this was their entire intent was, is there a way to attack this device by phoning into it from the outside? So they took a disassembly of it using their debugging tools, started reverse engineering it, figured out what was going on.

The normal faxes are two-tone black-and-white. They contain essentially a big TIFF, T-I-F-F, format image, although during the handshaking phase the various metadata for the following TIFF image is exchanged. They looked there. They couldn't really see any way of compromising it. And the TIFF interpreter itself looked solid. Again, no obvious means of exploit. But they noticed in the initial handshake there are many, you know, fax protocol, I think that's T.30, has been around for decades. And it has evolved. This machine, as do the HP machines, also advertises that it can accept a JPEG image. It says it supports color, maybe just half-tones in order to be accommodating. But there is a capabilities bit saying can accept a JPEG.

Well, it turns out that in the standard, if you're sending a JPEG, then the metadata for the JPEG is included in the JPEG, not in the handshaking. They looked there, and they found, I mean, like it was a - I'm speechless. I mean, there were so many problems with what they found. They immediately found a number of obvious buffer overruns. They were able to basically pick and choose which ones they wanted for the ease of their own use. And they found one that gave them maximum flexibility and freedom, where basically they worked out how they could dial up the fax machine unmodified; right?

So, I mean, all of that hacking that I talked about was information gathering to reverse engineer what was there. And this is what HP has reused universally, that is, that core across all of their combo devices. And I don't know if I could say there are hundreds of them. But, I mean, the list is endless, over time, of all the devices which are vulnerable. So what they designed was a "fax" which, when the fax machine answers, they do the little handshake and agree, and then they upload this JPEG image which compromises

the JPEG interpreter, allows them to execute a buffer overrun. And because they wanted them, I mean, it wasn't just enough to get in the fax machine. They wanted to demonstrate true capability. It then loads the EternalBlue and DoublePulsar NSA weaponized attacks to turn around and go out into the network to which this device is attached and take over all the machines there.

So my point is that today this is all public knowledge. Their write-up, I mean, it stops short of providing the binary code to do this. But it's just - and it took good hacking. I mean, people needed to know how to do this. But when you consider how target-rich the environment is, as I mentioned earlier, Google lists the fax numbers of 300 million machines. All Japanese businesses apparently have fax numbers; 45% of Japanese households do. Again, we don't know that they're HP. They may be Canon and probably are, other Japanese brands. On the other hand, we don't know that they're not equally affected. It was only HP that these guys attacked. They responsibly disclosed. HP said, you know, I'm sure there are people not sleeping. It was like, OMG, immediately produced, across the board, like every single device has firmware updates.

I updated my own machine this morning, thus the Picture of the Week is that picture. I tweeted it immediately, a link to the HP site. If you just go to printers, there's drivers and downloads. There's a firmware tab. You tell it the device you have. It'll point you at an EXE. You download it and run it. It'll find the device on your network and patch it. Interestingly, I wanted to know whether it would patch it if it had already been patched, and so I ran it twice. The second time took longer, but finally finished, but it didn't tell me that it had already been patched.

So somebody who's cautious may want to work through the devices, either their built-in network monitor to get the current firmware version, or maybe it shows if you dig around in the UI. If you have an LCD screen on yours, find out what the firmware is first, then run the patch, then confirm. Essentially, this is the first occurrence of a through-the-phone attack, that is, that turns around and attacks a state-of-the-art network to which this phone-connected fax printer is attached in order to do its job. HP has responded. It's very likely that other manufacturers are vulnerable. And anyway, so that's really the gist of this.

They had sort of a fun Q&A in their coverage, and I've got the links for all of this in the show notes. They asked themselves the question: "What is this research about? Check Point Research has uncovered critical vulnerabilities in the fax protocol." Now, that's not true. It's HP's implementation of the fax protocol. They say: "These vulnerabilities allow an attacker with mere access to a phone line and a fax number to attack its victim's all-in-one printer" - in this case HP that hasn't been updated - "allowing him full control over the all-in-one printer and possibly the entire network it is connected to."

They then ask: "Does this apply to all all-in-one printers? No. We conducted our research on all-in-one fax printers. However, similar vulnerabilities are likely to be found in other fax implementations such as fax-to-mail services, standalone fax machines, et cetera." Then they ask: "Who uses fax anyway?" Much as you did, Leo. "Surprisingly," they answer, "fax is still used by many industries, governments, and individuals around the world. These include the healthcare industry, legal, banking, and commercial, some of which are governed by regulations, and others simply for legacy reasons."

They ask: "What is the severity level of this vulnerability? HP classified this vulnerability as 'Critical.'" In fact, yes, 9.8 out of 10. So as bad as it gets. "How does this affect organizations and consumers?" They answer: "Once an all-in-one printer has been compromised, anything is possible. It could be used to infiltrate an organization's or consumer's internal network, steal printed documents, mine bitcoin, or practically anything."

They ask: "Does this apply to all fax machines? Our research was done on HP Officejet all-in-one printers, though this was merely a test case. We strongly believe that similar vulnerabilities apply to other fax vendors, too, as this research concerns the fax communication protocols in general." Again, no. It concerns the implementation of the fax communication protocols. There's nothing that the fax communication protocol itself does that allows this to happen. It just wasn't written right. But it's difficult to write it right.

"Is it widespread?" they ask. "By our estimates," they say, "there are currently hundreds of millions of fax machines still in use around the world. Financial reports from Wall Street indicate that tens of millions of all-in-one printers are sold worldwide each year." So it hasn't gone away. Still very popular. "Has it been fixed? We worked closely with HP to fix the vulnerability; and, following the process of responsible disclosure, they managed to release a patch before this publication. In fact, if your device is already configured to auto-update," as Joel's was, I mentioned before, "then the patch has likely already been applied," as he found.

"This patch, however, only applies to HP all-in-one printers, and the vulnerability may well still apply to devices from other manufacturers, as well." So just again, props to HP for beginning to have auto-updating devices. We need anything that is autonomous like a printer to either alert people that it needs patching or just to do it itself in the middle of the night when no one is using it, like our routers should be.

"What should I do to protect myself? If you own an HP Officejet all-in-one printer, then follow the instructions from HP." They provide a link in their Q&A. I have a link in the show notes. I tweeted it this morning. "In addition, you should implement segmentation policies, software patching, and proper IT hygiene. Please see our 'Recommendations' section," they say, "in the blog post above. If you are no longer actually using the fax functionality in your all-in-one printer, then we recommend you to disconnect the phone line."

Perfect advice. You know? Yes. Just unplug the phone line. Or as you said, Leo, only plug it in when you need to send or you're expecting to receive a fax. That would work, too, as an interim. And again, non-HP combo, like connected fax devices, hopefully all those companies are working on this, too. So do keep a lookout for your non-HP device updates. And in the meantime you may want to take it off the network or disconnect it from the phone, rather than just leave it sitting there.

And then they ask, finally: "Has it been seen it the wild?" And they answer: "Not yet. Our research was intended to highlight a potential security risk." And again, this isn't the sort of thing that I would expect to see. You know, you're not going to get a botnet created from it. It's not going to be high visibility. It's going to be an attack in the dead of night by somebody who develops this, who weaponizes this, and then uses it to get into a company whose device is old, doesn't auto-update, isn't aware of this. Their IT team, maybe they don't even have one, or they just aren't fixing this. But I just think this is probably the most significant vulnerability we've seen so far this year. Spectre and Meltdown, they were bad. They were theoretical, difficult to do.

Leo: Still not in the wild; right? I don't...

Steve: Exactly. And no practical attack has yet been seen. This one, boy. I mean, I...

Leo: It's already in use, frankly.

Steve: Yeah. There is just - it is too tempting and too easy using, you know, off-the-shelf standardized tools for bad guys to do this, to follow in the footsteps of Check Point who, in their blog post, I mean, they explained it all. They've got screenshots and code, and they show what they found. I mean, it's complete documentation of what HP had that was vulnerable in their machines. And devices that don't get updated are just going to be sitting there, waiting for the phone call.

Leo: Wow. My friend, we have come to the end of this fine rendezvous. Once again, you've set our minds at ease - not. Not. Actually, for a post-DEF CON/Black Hat episode, this wasn't too bad. Too bad. You didn't talk at all about voting machines. Now, that's scaring me.

Steve: Yeah.

Leo: Ladies and gentlemen, we invite you to join us for the live edition of this show. We actually stream all the shows as we make them. So it's a little different than the final product. But it's a chance for you to kind of watch along with the other folks in the chatroom and comment. We do the show every Tuesday, 1:30 Pacific, that's 4:30 Eastern, 20:30 UTC. And you can watch it stream at TWiT.tv/live. If you do that, join the chatroom, irc.twit.tv. They're all watching the show, too, and talking about it, commenting on it. If you can't watch live or listen live, you can always get on-demand versions.

Now, Steve hosts the audio at his site, GRC.com. He also has - he's the one-and-only source for transcripts, which usually take a few days for Elaine to do, but they come out shortly after the show. And a lot of people like to read along. It's also a great way to search. You can search the transcripts on his site and find exactly what you want from any one of our 676 episodes, 13 years of Security Now!, GRC.com. While you're there, pick up a copy of SpinRite, the world's best hard drive maintenance and recovery utility. You might also want to check out some of the other free stuff Steve does, catch up on SQRL, Perfect Paper Passwords, and more. GRC.com. Steve himself is on the Twitter at @SGgrc and takes questions and comments and tips there. You can DM him.

I have audio and video of the show in case you wish to see Steve's moustache in person. Yeah, no, it's good. It's a good moustache. Very high-quality moustache. Now everybody's going to want to download the video to see it. Or get the mug. You can actually get the moustache mug. So our audio and video versions are at TWiT.tv/sn for Security Now!. The mug is at TWiT.tv/store. Have your very own Steve Gibson mug. Let the world know you're secure. And let's see what else. Oh, if you don't yet subscribe to the show, my recommendation is subscribe, you know, go to your favorite podcast app and search for Security Now!, and then press the Subscribe button. That way you will get it automatically. And that's...

Steve: And also I just will say I love hearing people's SpinRite success stories. We've been sharing them for 13 years now.

Leo: We need more, yeah.

Steve: So GRC.com/feedback, or you can tweet me at @SGgrc. Share your success, and I will share it with our listeners.

Leo: Yeah.

Steve: Since every one just really - I'm so happy that we're able to help people.

Leo: Okay.

Steve: And next week Year 14 begins. Woohoo!

Leo: Wow. That's really kind of mind-boggling. A kid who was born when we started the show would be a teenager now. That's mindboggling, Steve. All right, Steve. Have a great week. See you next time.

Steve: Talk to you next week. Bye.

Leo: Very nice. Very nice.