SECURITY NOW!

Transcript of Episode #663

Ultra-Clever Attacks

Description: This week we will examine two incredibly clever, new, and bad attacks named eFail and Throwhammer. But first we catch up on the rest of the past week's security and privacy news, including the evolution of UPnProxy, a worrisome flaw discovered in a very popular web development platform, the first anniversary of EternalBlue, the exploitation of those GPON routers, this week's disgusting security headshaker, a summary of the RSA Conference's security practices survey, the appearance of persistent IoT malware, a significant misconception about hard drive failure, an interesting bit of listener feedback, and then a look at two VERY clever new attacks.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-663.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-663-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here with all the week's news plus a couple of clever attacks - cleverly named, as well. One's called eFail, and it really is an issue if you use PGP or S/MIME to encrypt your email. The other's called Throwhammer, and I'll let Steve explain how that works. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 663, recorded Tuesday, May 15th, 2018: Ultra-Clever Attacks.

It's time for Security Now!, the show where we cover your security and privacy online with this guy right here, Mr. Steven Gibson of GRC.

Steve Gibson: And look, I'm lit up just exactly right today, Leo. I'm not too bright and not too dim.

Leo: Has it been a problem in the past? You've always seemed just right to me.

Steve: With my brightness?

Leo: Yeah.

Steve: See, it's bright.

Leo: Super bright now. What, did you get special lights?

Steve: No, no. It's just that your screen behind you is sometimes - it changes.

Leo: Oh, that shot. Well, no, yeah, that's not a - yeah. No, I changed your brightness on my end. So we've got a big one today, I think.

Steve: Well, yes. Two very clever, I mean, okay. Let me finish the sentence. Two very clever attacks, one against encrypted email in general, so it affects the two primary encrypted mail technologies, PGP and Secure MIME, S/MIME. And it's not a flaw in either of them except that it exploits a weakness in the security protocol. But it's just sublimely clever. And as I was reading into it, and I realized what these guys had figured out, it was like, oh, that's just so - that's just, like, whoa. So I thought, okay, I just have to explain what they did because it just - it's a toe curler.

And then, as we always have said, attacks don't get worse, they only get better, quoting Schneier, I think. Bruce was the first person to say that. We have the inevitable evolution of Rowhammer that we were just talking about a few weeks ago, in fact it was last week, about the use of GPUs to induce the Rowhammer attack because they're lightly, if at all, cached, unlike CPUs that have sometimes multiple levels of cache so you have to do all kinds of crazy cache avoidance in order to get down to the memory.

So now we have Throwhammer, which as its name suggests is remote Rowhammer. And this is very worrisome because, as we've often said when we're trying to sort of place attacks into a fair context, well, yes, it's bad, but you've got to have bad guys' code on the machine in order to be pounding on the RAM to get some sort of leakage. Well, not anymore. Now you can do it over the 'Net. So that's Throwhammer. So we will talk about both of these very clever attacks.

But first we've got to catch up, of course, on the rest of the week's security and privacy news. We've got the evolution of UPnProxy. That was the use of Universal Plug and Play that Akamai discovered and we reported a few weeks ago. Well, it's already evolved. We have a worrisome flaw discovered in a very popular web development platform that's probably going to get exploited. It hasn't yet, but inevitably.

Three days ago was the first anniversary of the revelation of EternalBlue, which the WannaCry cryptomalware used to such devastating ends, and then several other malwares adopted. We're one year downstream, and we've got sort of a chilling graph of the evolution in the use of EternalBlue. We also have those GPON routers that were found to be very insecure and exploitable. There are now five botnets fighting over them. We have this week's disgusting security headshaker. It's like, okay, what year is this, and we're still making incredibly dumb mistakes?

We have an interesting summary of the recently completed RSA Conference's security practices survey, where all of the attendees of the conference were asked to, here, fill out this survey of some interesting and somewhat chilling results that we need to share. The appearance of the first persistent IoT malware. Traditionally it's been possible just to unplug your light bulb because the malware was running in RAM. Well, that's not true anymore.

Also, I encountered a significant misconception about hard drive failure, relative of course to the use of SpinRite, that I wanted to briefly address. And that is the misbelief that a failing hard drive needs to be replaced. And it's not that that's not true, it's that hard drives can have problems when they're not failing. And so I want to dig into that a little bit. I have an interesting bit of listener feedback. And then, if we're still on Tuesday, we'll take a look at these two very clever attacks. So, yeah, I think another jam-packed podcast, as you suggested.

Leo: I'm really curious about the PGP attack.

Steve: Oh, Leo, it's just sublime. Oh, my god, it's so cool. It's brilliant. I can't remember what Matt termed - Matthew did a series of tweets, and he called it - I'm scrolling down here through it, he called it "a masterpiece in exploiting bad crypto." He said, "It's an extremely cool attack and kind of a masterpiece in exploiting bad crypto." Oh, and, yeah. So, yeah, I think everyone's going to get a kick out of it.

Leo: Good, good.

Steve: So our Picture of the Week was one that I've had in my picture file. It's not apropos of anything in particular this week. But I got a kick out of it because it does help to sort of, you know, there are a lot of people who are chafing at the idea of HTTPS. We know you cannot have logged-on sessions using browser cookies, web cookies, which is the way you maintain session state these days, unless you have privacy in your communications. Otherwise, anybody who's able to passively sniff traffic, as you can, for example, in an open WiFi situation, is able to impersonate you by grabbing that session state. So HTTPS is where we're headed.

It happens that the listener feedback that we have today argues against the idea that HTTP is dead forever. And I got a kick out of this because it's a cartoon showing the tradition snake oil salesman telling you - holding up a bottle, saying that this special elixir will cure all of your ills. And so on the tree it says, "Dr. Marvel's amazing HTTPS, guaranteed to cure all website security problems except: plaintext passwords, SQL code injection, buffer overruns, social engineering, malware, spyware, adware, ransomware, trojans, CMS hacks, cross-site scripting, foreign URL injection, Flash exploits, Acrobat exploits, Java exploits."

Leo: Otherwise, it's great.

Steve: Otherwise, boy, this thing will fix everything that ails you. So, yes. I liked that because it's a nice counterpoint to everyone rushing to secure everything. I mean, we have to have that, too. But by no means is it the cure-all for all of our problems with the web. It just means that a class of attacks are much harder to perpetrate, and there's still lots of work left to be done. So pretty much like health in general, it's a large surface, a large attack surface.

So, okay. I did want to follow up on last week's warning or caution or "watch for it" note about Spectre. It was, as we know, Spectre NextGen, where the news is - and this came from Heise over in Germany and had been well researched and followed up and verified -

that there are eight new problems of completely unknown type, one of which Google's Project Zero found; one of which is a biggie, and we don't know if that was the one that Project Zero found or not. We know very little about this.

So I just wanted to say we don't know anything more than we did. So I'll keep looking, but so far this is a mystery. Certainly at some point we will know a lot more. And the teaser, I think, is that the big problem is bigger than Spectre ever was. And that is either version one or version two of the original Spectre exploits. Apparently what's been found is something substantially easier to exploit, which the reporting says really does represent a problem for a shared hosting environment.

So one of the bits that came out of the RSA survey, the people that were there, we'll get to that in detail a little bit later, is that three quarters of the attendees are in companies who use cloud providers - Google, AWS, Microsoft. So three quarters of enterprises of the attendees of the most recent RSA Conference are using cloud services. And cloud services are not the only, but certainly the biggest worry and the biggest target for this kind of exploit, for something that allows you to break out of a VM containment or any kind of a sandbox or process.

Okay. So we talked about Akamai's announcement/discovery that there are a lot of consumer routers with Universal Plug and Play port, the configuration port, or maybe the file it turns out, exposed to the Internet; and that, as a consequence - and we put it in the show notes when we talked about this a week or two ago, there's that classic sort of spy Internet, like where the bad guy bounces their traffic off like all around the world nine times or 19 times, however many, in order to defy the authorities' ability to track them down, and then finally land somewhere. Which was always a bit farfetched because that suggested there were, like, all of these insecure systems everywhere.

Well, turns out that with this ubiquitous presence of consumer routers on LANs, which of course I've been championing forever because it is, when properly done, it is a really good hardware firewall. I mean, you absolutely want that. I can't imagine putting just a computer right on the Internet ever without having the NAT, the stateful NAT layer which drops unsolicited incoming packets. Well, unfortunately, with this crazy popularity of these routers, they're not all being done right.

And so what we first learned from Akamai is that the routers' NAT mapping, which Universal Plug and Play is able to manipulate deliberately, the routers do not check that the IP on either side of the packet are on either side of the router, meaning the only valid mapping that that translation table should provide is from LAN to WAN. There is never a place, well, there's almost never a place for LAN to LAN, although there are some instances where you can reflect from the router back into the LAN. But certainly never from WAN to WAN, from the Wide Area Network, the Internet, out to the Wide Area Network. Turns out there are routers that don't check that, which makes them a perfect target for somebody who wants to hide their traffic as botnets and various attackers want to. So that was the first instance.

What the Imperva security firm has since discovered is that it is possible, and it is actually being done, that the ports are being changed in addition to the IPs. Now, that's not a surprise. NAT has to do that. That is, technically it's NAPT, that is, not just Network Address Translation, but Network Address Port Translation, because ports are almost always being changed also because the NAT table needs to use additional information for packets coming back to know where they're going. If three people on the LAN were all going to Amazon, then when the packets come back, they're all coming back from Amazon. So the destination port number to the router is the way the router knows which one of, in this example, those three people on the LAN that packet should go to because you just need additional bits in order to encode the identity of the machine behind the LAN. So ports have always been changeable.

What the bad guys have figured out is that one of the easiest DDoS mitigations, the Distributed Denial of Service, where for example you're using a bandwidth amplification attack with DNS or network time, either DNS or NTP, is that the source address of the packets will be that of the service, like 53 or 123, respectively, for DNS and network time. So one of the easiest things that a DDoS filter can do, a simple way of mitigating these attacks, is simply blocking all incoming port 53 traffic, or all incoming 123 traffic. That is, it's trivial to do that at the border of the network and just drop all the traffic.

So because the bad guys are clever, they said, hey, this is another thing we can do using our NAT port translation, is we can enlist the services of these accessible NAT routers all over the Internet - and there's apparently on the order of 1.3 million of them available - in order to translate the port number. And so the idea is they will use the DNS amplification in order to create additional traffic, then have that bounce through one of these NAT proxies and bounce off of the proxy so the traffic stays public, but changes the source port to something other than 53 or 123 in order to bypass static DDoS filters, which many networks now have in place. So yet another way of using the unsecure and insecure technology in order to create additional traffic.

And specifically, and this is what was worrisome, what Imperva found is that many of these insecure routers actually have the XML file which describes the current mapping, its "rootDesc.xml," is publicly available. Shodan has been searched for the presence of that file, and 1.3 million results were found. So you remotely massage that file using the Universal Plug and Play protocol, or directly using that file, and are able to essentially turn these 1.3 million consumer routers into little switchyards, bouncing traffic off of them, changing IPs, changing ports, and getting up to a lot more mischief just due to the numbers which are available, and the fact that increasingly these are all very well connected devices. So again, attacks only get better.

Also I mentioned a very popular app development platform having a problem. There's a platform known as Electron which hadn't crossed my radar before because I'm about as far away from developing apps on a web platform, since I'm using assembly language, as you could get. But Microsoft's Skype, Visual Studio Code, GitHub's Atom code editor, the Brave browser, including well-known desktop apps for services such as Signal, Twitch, Discord, Basecamp, Slack, Ghost, and even WordPress, that, is the desktop services for those are all written as a web app on top of this Electron app development platform.

Leo: Yeah, it's really common. It's basically Chrome.

Steve: Right, right.

Leo: That's the negative is you have to bundle a copy of Chrome with every app. It makes the apps quite large, which is, I know, anathema to you.

Steve: Yeah, exactly. Well, and what's worrisome, Leo, is that the developers understood that the Node.js library, while powerful and deep, is also dangerous to have on the desktop. So by default there's a setting in the configuration file. There's a web preferences config file that has node integration set to "false," which blocks access to the Node.js APIs. I'm sorry about - can you hear this background noise?

Leo: Yeah, a little bit. It's not bad. It's all right.

Steve: Okay, good. I've got the microphone turned away from it. They're grinding up some trees outside.

Leo: It's mild. I hear every once in a while [mimics sound]. It's like "Fargo."

Steve: Yes, it's not my stomach growling. Okay. So they've got node integration set to "false." But a security researcher, Brendan Scarvell with Trustwave, discovered that it's possible to flip node integration to "true." This can occur if another setting in the file, webviewTag, which is normally set to "false," has not been explicitly declared. In that case it's possible to use a cross-site scripting instance to create an instance which has node integration set to true, and then be able to run an attacker's code on the machine where this, as you said, Chrome essentially HTML JavaScript-based system is present.

He discovered this in March. He privately reported the flaw to the developers, who immediately fixed it. He now has finished, published proof-of-concept code which allows an attacker to exploit any cross-site scripting flaw that may exist in one of these applications to extend, essentially give them access to the underlying OS and run their own code. And remember that cross-site scripting is only - the only thing required is that something that an attacker provides is shown on the page. That is, that's the trick is you just want to get something that is unfiltered, that has some HTML tags in it that are not being properly encoded. They will be interpreted by the JavaScript in order to create the exploit, the very common cross-site scripting.

So anyway, the problem is that this has been fixed. It's been fixed for several months. Yet there are - and if you look at this, at the /apps, electronjs.org/apps, it is very, I mean, it's incredible how many apps have been written in this. Not just these mainstream big guys, but lots of other apps are written on the top of this platform, probably because, as you said, it makes it very easy to move something from a web page over to the desktop using this development platform.

So the problem is that, in the long term, if these things are not fixed, if people are not keeping them up to date, even if their publishers keep them up to date, which isn't clear will happen, this creates a vulnerability that could be exploited for a long time to come. And, yeah, you have the list there. And, I mean, just it scrolls on and on and on and on. Many are obscure little special purpose things. But also the big guys use it, as well.

Leo: Yeah, I mean, it's a very popular web framework. Probably not the best. It's certainly, you know, fat. But it probably gives you all the functionality you want because you've got a browser in there. You know how to do CSS and JavaScript, you're set.

Steve: Exactly. Yup. And unfortunately you don't want to let things get to the underlying desktop unless it's on purpose. And this provides...

Leo: So I'm surprised it's not - WhatsApp uses it. I'm surprised it's not sandboxed.

Steve: Well, they deliberately kept it Node.js disabled because there are so many ways that does give you access to the underlying OS. No, it's just it's a very powerful library.

Leo: Signal uses it at the desktop, yeah.

Steve: Yes, yes, Signal. So anyway, I hope that the developers will fix their apps, and people using the apps will keep them current. What he found is that essentially any vulnerable version of Electron, less than 1.7.13, 1.8.4, or 2.0.0 beta 3 - so they apparently fixed it even after, I guess just before 2.0 got finalized. Anything younger than that, that has apps built on those, would be vulnerable. And it's not clear whether you could change the framework underneath the app. I think that you can't, that it's all bound together. And so you can't just replace one of the pieces. You probably have to get a new updated whole app kit from the publisher.

So, okay. Under the topic of old flaws never die, which of course we see all the time, three days ago was the first anniversary of WannaCry. Leo, doesn't time fly.

Leo: Wow, I can't believe it, yeah.

Steve: So, and WannaCry was, I mean, its major innovation was that it used and leveraged the EternalBlue exploit, which was believed to have been developed by, come from, and leaked from the U.S. NSA. And of course it powered the spread of WannaCry, NotPetya, and Bad Rabbit malware. Then it kind of got quiet. What was interesting is that, whereas the original EternalBlue only worked with XP, Windows 7, and the equivalent server platform, Server 2008 R2, the underlying flaw in SMBv1, which is what this used, has since then also been made to work under Windows 8, the equivalent Server 2012, and even Windows 10. This essentially broadening to the entire Windows platform hugely increased the exploit's ability to infect and has basically turned it into a commodity among malware authors.

I have a chart from ESET which shows over the last year, so here is a one-year period of time, essentially the lifecycle so far of the EternalBlue exploit. WannaCry itself is still active and attempting to find and infect anything that comes online publicly, any systems that come online publicly. And so we can see here, you have it onscreen now, a big jump in its activity. Then it got blocked and kind of went silent. But then, as other platforms have come onstream, and as it's been moved into sort of the default toolkit for exploitation, it's continued to grow and is beginning to approach the original - the exploitation level of activity. So just as Code Red and Nimda continue their search for new victims, WannaCry itself is active. But the EternalBlue exploit is being leveraged by an increasing breadth of malware in general.

We talked about these GPON routers that were the optical fiber-based routers being offered by several ISPs. Naturally, the news got out. Botnets that were active decided to increase their own activity and jump over to those. So now, and you'll get a kick out of this, we have the Hajime botnet that we talked about.

Leo: Hajime.

Steve: Hajime, Mettle, and of course Mirai that has been a lot in the news; one that we haven't talked about, Muhstik; and then also the Satori botnet. So we've talked about Hajime, Mirai, and Satori, three of those five in the past. Well, they're now - they've switched their attention to these GPON routers. And that's really what we're sort of seeing now is that botnets are being taken seriously. They're being maintained and updated. And at this time it looks like something just shy of one quarter million routers, that is, there are one quarter million GPON routers. And when I say "shy," it looks like it's about 240,000 which are vulnerable.

The firm vpnMentor was the discoverer of this GPON vulnerability. And unfortunately the ISP is very uncooperative. They're still dragging their heels, denying that there are that many vulnerable routers still present, even though people are scanning for them and infecting them. I mean, there's botnets running on them. So they have an unofficial antidote patch. I don't know whether any of our listeners have GPON routers. But it's clever. They have a means of patching the router where - and I've got the link here in the show notes. It's www.vpnmentor.com. Under their Tools directory is a GPON router antidote patch. They caution that it is not official; that of course any official patch needs to come from the vendor and should. But this vendor is not looking like it's in any hurry.

And Leo, if you scroll down, you'll see a field that you fill in. Essentially, you give them the internal IP of your gateway and the Telnet password that would allow you to log onto your router. So what happens is that brings up a page on your browser which is now inside your LAN. It runs script which connects to your router's Telnet interface on the LAN side, uses the password you gave it in order to then load a patch onto the router. And I'm sure afterwards they tell you to change the password of Telnet.

Leo: And you've verified this is entirely safe and completely okay.

Steve: Well, again, at this point, if somebody had a GPON, one of these vulnerable GPON routers...

Leo: You're no worse off, are you, I guess.

Steve: Exactly. You're in bad shape already. Your ISP is apparently...

Leo: They didn't really - they didn't need you to do this. They could do it themselves.

Steve: Exactly. Exactly. So probably the best course of action would be to say, ah, yeah. And these guys appear to have their hearts in the right place. So, yeah. To the degree we can trust anyone, these guys seem about as trustworthy as you could get. Or, again, what choice do you have?

Leo: This wood chipper's getting louder.

Steve: I know.

Leo: Are they getting closer?

Steve: I think the trunks are getting larger.

Leo: That's probably right. They were doing the little skinny branches. Now they're doing the big boys. Well, we'll just live with it. It's fine.

Steve: Yeah. So in this week's, as I described it at the top of the podcast, this week's shocking insecurity headshaker, it's like, what year is this? Is this 1995? Bleeping Computer reported the news of, get this, 5,000 routers with no Telnet password, exposed to the public. It's just incredible.

Researchers with NewSky Security, which is a cybersecurity company specializing in IoT security, discovered that the exposed devices were Datacom routers from a Brazilian ISP, Oi Internet, which Oi Internet had provided to their customers. There were three of these Datacom routers, DM991CR, DM706CR, and DM991CS. They were found to have blank Telnet authentication, with the Telnet port wide open to the Internet, accepting all comers. The researcher who discussed this with Bleeping Computer's reporter said that the routers' manuals clearly indicate that the devices are designed to come with a passwordless enabled Telnet service by default, and that users are then expected to configure the password for themselves.

So I'm just, like, how does this happen in 2018 that, first of all, a manufacturer can offer a router with an unconfigured Telnet service running on the WAN interface with no password? How does it happen? That's just - I'm stunned that that could be the case. But I just, you know, I guess maybe they weren't meant to be consumer routers; or the presumption is, because they were OEMed, the presumption was that any OEM would of course turn the service off or lock down, just turn Telnet off completely on the WAN side, or provide a good username and password. I just, again, somehow, some miscommunication. But, wow. Unbelievable that, I mean, imagine buying a new router which no consumer has any idea what to do with. They just plug it in and assume it's an appliance. Yet it's got Telnet wide open. Wow.

Okay. One more, then we'll take our second break. I have a link here to the PDF that I put a copy of on GRC's server since you had to go through some rigmarole in order to get it, and I didn't want to ask all of our listeners to do that. And it's not behind a pay wall or anything. And they've got all of their advertising all over it. This was the highlights of the security form from the recent RSA security conference. And I got a kick. Down lower are a bunch of pie charts. But get this. Only 47% of organizations patch vulnerabilities as soon as they are known, so less than half of organizations patch immediately; 16% wait for one month; while 8% admitted to only applying patches once or twice a year.

So, I mean, this is good news for bad guys. And these are people attending the RSA Conference. You could argue that this is not even a representative cross-section of enterprises. This is RSA security conference attendees saying, eh, yeah, you know, half of them don't do them as soon as they're known; 16% wait a month; 8%, eh, we get around to it once or twice a year. Wow.

Okay. Also 16% of organizations have admitted - now, this is admission on a survey at RSA - 16% of organizations have ignored a critical security flaw because they didn't have the skills to rectify it; while 26%, okay, one quarter have ignored a critical security flaw because they didn't have time to fix it. It's like, yeah, okay. Critical, but we're busy. We're busy here. One in four firms.

When asked what route they would take to hack their own companies - and I thought this was a really great question. When asked what they would do to hack their own companies, 21% said they would enter through the company's public cloud hosted computing; while 34%, so one in three, said they would use social engineering if asked to hack their own company. When asked if they expected that attack would be successful, 71% said it was likely or highly likely that they could succeed from outside hacking into their own company. Only 9% said it was very unlikely their attack would succeed. And this is where I also mentioned that three out of four of the organizations use a commercial cloud provider as part of their infrastructure.

And then, finally, only 17% of organizations - and these tend to be sizable, you know, they're RSA attendees - have ever hired a penetration tester, a pen tester, to assess, objectively externally assess the security of their own networks. So of those 17%, 46% found a critical flaw which could have put their organization at risk. Which I would imagine our listeners do not find surprising. However, 35% believe that, if they were to hire a pen testing service, although they haven't, they would not surface any new risks. So to me that sounds like CIO or CTO hubris, where they're like, oh, no. I'm in charge of security. We haven't hired anybody. And if they did, it would just be a waste of time and money because we're secure. Right.

Leo: They've chipped all the trees.

Steve: I think they're done. They've ground it up into dust.

Leo: Suddenly the view is of crystal clear skies.

Steve: So BitDefender Labs has identified the first persistent IoT malware. We've talked many times about how these infections just live in our routers' RAM, and that just a power down and reset, reboot, power back up, whatever, is enough to clear it out. Well, turns out that was then. BitDefender Labs has found what they call the "Hide and Seek" botnet. They first discovered it earlier this year and have had their eyes on it, tracking its evolution and progress. It's infected close to 90,000 unique devices, not only routers, apparently IPTV cameras. And as I mentioned, what we're seeing now is just much more sophistication in botnet seriousness than previously.

One of the things that caught their eye is that this botnet establishes a peer-to-peer command-and-control network using UDP with a fully custom homegrown peer-to-peer protocol. So this creates a mesh of connected botnets which creates much more connectivity than we normally see. Normally, as we know, botnets all refer back to some command-and-control server somewhere, which makes them easy to take down, just by taking that one server offline. Suddenly they're all sitting around waiting to get instructions, and none are forthcoming. Here, by creating a mesh of interconnected botnets, all you have to do is talk to any one, and this will propagate the command-and-control system throughout that highly interconnected mesh, which makes it virtually impossible to take this down. So this is a big step forward.

Also, unlike the previous botnets which lived in RAM, this one uses the various brute-forcing of Telnet access to get root. And now, even though previous infections, various malware might have been able to get root, it wasn't using the root privilege to write itself into persistent storage. This Hide and Seek botnet does. It gets into Telnet using brute force, identifying devices using a bunch of well-known default admin usernames and passwords. And we've talked about how effective those can be in the past. Once it has that, it copies itself into the /etc/init.d directory, so it gets run whenever this device reboots. And this includes routers and IPTVs. It also has 10 different binaries compiled for various platforms, including x86, x64, the 32- and 16-bit Intel chips, ARM, both Little Endian and Big Endian, SuperH, and PowerPC, among others.

So a lot of work has gone into putting this thing together. And as I said, it's now installing itself permanently into the firmware of the devices it infects and establishing a persistent and very hard to kill UDP communication mesh between it and all the others. Oh, and when it gets installed, it then begins scanning its neighborhood for other infected botnets. It also closes the door behind it so that nobody else, no other malware is able to follow it in through the Telnet port in order to infect that device. So, I mean, this is what you would design if you were very competent, and you wanted to create a persistent malware network out on the public Internet. And it actually exists. It's not science fiction.

Okay. As I mentioned at the top of the show, I wanted to address a misconception. And I want to thank, it looks like Ely Riggs in Tallahassee. And the subject was VCC Citi, and I didn't know if that was an abbreviation for Vatican because of course Father Robert shared with us the fact that SpinRite was now being used at the Vatican for...

Leo: Oh, that's neat. I didn't hear that. That's great.

Steve: ...recording hard drives, yeah. He left his copy there with an associate, and it's been blessed now. We had made it up to the altitude of the International Space Station. Now we've made it up to an even higher altitude.

Leo: I think it's even higher, yes, yes. That's great.

Steve: So anyway, so this person writes, "Hey, Steverino. Great show, as always. Your SpinRite testimonial was incomplete. After repairing an unbootable drive," and then we have in all caps shouting, "BUY A NEW HARD DRIVE, or upgrade to an SSD. Do not continue using a failing hard drive."

So I wanted to take a minute to address that because, first of all, I absolutely agree you should not continue using a failing hard drive. But an unbootable drive, or a drive that SpinRite is able to repair, doesn't mean the drive is failing. We need to remember, first of all, drives have a huge backup of spare sectors. Why do they have spare sectors? Because in the normal course of their service life, they're expecting sectors to fail. And when sectors fail, the drive says whoops and essentially takes that sector out of service, moving that sector's data into one of the spares in the pool of spares. Sectors fail over time just through the actual - the head flying over the surface is flying very closely, and there is a tiny mechanical stress that the head and the surface platter put on each other.

So there's a little bit of flexure; and, over time, that can evolve a defect. What normally happens is the drive will see that there's a problem forming. But again, remember that drives are performing error correction now all the time. And the use of ECC used to mean, oh, my god, thank goodness we're able to recover the data from the sector. Stop using the sector. Now ECC is, yeah, okay, fine, we corrected another error. The idea is that...

Leo: Just keep on going.

Steve: Yeah, yeah, exactly, exactly.

Leo: I would imagine that some of these 4- and 8-terabyte drives are probably getting ECC failures pretty constantly; yeah?

Steve: Yes. And in fact that's one of the things that SpinRite shows is the rate of error correction. It's not zero. It's startling. And so what really happens is when the rate of error correction begins to increase, that's an indication that you've got a problem with your drive. But what's so cool about ECC is that what the error correction math creates through this cool ECC technology, it creates an XOR bitmask which is a pattern of ones where the bits are wrong, and an offset for where in the sector's typically 4,096 bits, where the mask should be applied in order to flip the wrong bits to make them right. That's called the "syndrome." And so naturally the first bit of the syndrome is a one, and the last bit of the syndrome is a one. Otherwise you'd remove the zeroes, and then the first bit would be a one and the last bit would be a one.

The point is the distance between the first one and the last one is the number of bits in error. And the drive will watch that and happily correct them until some threshold because there's a limit to how long a run of error it's able to correct. So as the problem grows over time, at some point the drive says, uh-oh, we're past a threshold of comfort. So while I can still correct the sector, I'm going to correct it one last time and take it out of service. Now, that's the "everything going well" scenario.

Of course SpinRite is typically brought in when the "everything going well" scenario fails. And so that's where SpinRite comes in with the just give us the data one last time. Please, just once more. We'll never ask for it again. Just one more time. And it often does, in which case it and the drive breathe a huge sigh of relief, the sector is taken out of service, and the corrected data is relocated. The point being this isn't the failure of the drive. This is a sector that was allowed to get outside of the drive's correction tolerance band. But once fixed, everything is fine.

So it's, again, SSD is using and relying on error correction. Hard drives are using and critically relying on error correction. I mean, we couldn't have, I mean, there just aren't drives that aren't correcting errors now all the time. Some of them are soft. Some of them, if you retried, you would get a perfect read, except the drive wants to be fast rather than safe. It's biased in that direction. So it just goes ahead and says, okay, I had to correct the data. Maybe if I tried again I wouldn't. But I have correction able to do on the fly without slowing down, so I'm going to just keep going. So, you know, that's what's happening.

So anyway, I just sort of wanted to note that a failure that SpinRite corrects can just be a sort of a transient problem in one spot where something got a little bit out of control on the drive. But the drive has plenty of spares and plenty of service life remaining. SpinRite, again, running SpinRite from time to time keeps all of this from getting to the point where you may not be able to recover it.

Okay. And from Eric Paul in Chesapeake, Virginia, and this was the guy that I referred to at the top as his subject line is "Still need HTTP sites." And he wrote: "I just heard you on Security Now! that there is no longer a need for HTTP sites. I have an issue with that statement. At my house I have an old Netgear router that supports HTTPS access. But due to its age, it uses a self-signed SSL3 cert." And actually certs don't have a protocol. There's no such thing as an SSL3 cert. I'm assuming he means, for example, an SHA-1 cert, which is probably what he's referring to.

"Since modern browsers have decided that they will absolutely not allow access to SSL3 sites" - and actually he means SHA-1 signed certs - "I had to use an old version of IE" - that would do it, that is, SHA-1 - "which still allowed" - he says SSL3, he means SHA-1 - "certs to remove HTTPS-only access so that I can use a modern browser to access the router." Oh, I see. So he used an old version of IE to turn off HTTPS-only on his Netgear, then allowing him to use HTTP only on modern browsers. So that was nicely done, Eric.

He says: "Due to this reason, I had to convert both of my routers to HTTP-only access since I assume that at some time in the future all browsers will punish the self-signed certs used by both of those routers with no exceptions allowed." So I take his point. And I, and I imagine a bunch of our listeners, are beginning to encounter things like that, where this assumption by browsers and enforcement by browsers of the latest security standards are beginning to collide and clash with some of our older devices that aren't updateable and don't support the latest security protocols.

So anyway, I just sort of wanted to thank Eric for noting that. And I'll mention that I did mention that a recent Asus router of mine, I was delighted to see that it was obtaining its own Let's Encrypt certificate for itself. I thought that was a very cool use of online real-time certificate generation. But I have encountered myself older devices where it is necessary to jump through some hoops in order to access them because newer browsers just say no, or cough up a whole bunch of errors that you have to, like, fight your way through in order to still use what is older security. But in this instance, for example, there's zero danger in using HTTP to access the management of your own router on your own LAN. It's just like, you know, there's just nothing wrong with doing that. Yet the browser says no, no SHA-1 certs are allowed.

Okay. So the two ultra-clever attacks. And Leo, on the second page here - or no, it's the third page down - I have a picture that captures this that I will be explaining, but you're going to want to show that one when we get to what this is. So the first one is the PGP and S/MIME. Okay. So it's not a new vulnerability. It's a brilliantly clever leveraging of an original design flaw in encrypted email which affects encrypted email, thus PGP and S/MIME. The EFF wrote, and I'm quoting them: "Users are advised to disable email encryption plugins to avoid any attackers from recovering past encrypted emails after the paper's publication." And that did happen. It was supposed to happen today; but news leaked out, and so it was published yesterday.

The EFF wrote: "These steps are intended as a temporary conservative stopgap until the immediate risk of the exploit has passed and been mitigated against by the wider community." They said: "Users in dire need of using encryption to protect their communications channels are advised to use an instant messaging client that supports end-to-end encryption, the EFF recommended." And then in a follow-up the EFF said: "Not So Pretty: What You Need to Know About eFail and the PGP Flaw." And again, it's not a PGP flaw.

They wrote: "A group of researchers released a paper today" - this was yesterday - "that describes a new class of serious vulnerabilities in PGP (including GPG), the most popular email encryption standard. The new paper includes a proof-of-concept exploit that can allow an attacker to use the victim's own email client to decrypt previously acquired messages and return the decrypted content to the attacker without alerting the client. The proof of concept is only one implementation of this new type of attack." And that's the key. This is a new attack. This is not a new vulnerability. "And variants may follow in coming days." So the site where this is all put up for anyone who's interested is eFail, E-F-A-I-L dot D-E.

Okay. And so the ultra-short attention-getting version is an attacker who had previously obtained any encrypted email, that is, someone's encrypted email, through for example passive monitoring, or maybe by pulling from an encrypted stored repository.

Leo: Encrypted, not cleartext, the encrypted version.

Steve: Encrypted, yes. So if there's email sitting on a server somewhere, or on a cloud somewhere, which is encrypted so that only the user's client is able to see it, they are able through this incredibly clever, I mean, just, as I said, this is just going to - this is just amazingly cool, or as Matt Green called it, a "masterpiece" - can induce the recipient's, that is, the intended recipient's email client to decrypt and exfiltrate, that is, to send the decrypted content.

Okay. So a little more background first. Matthew Green, on May 14th, yesterday, tweeted a series of, like, nine tweets. He said, "New vulnerabilities" - and he gets it right - "in many GPG and S/MIME enabled email clients allows exfiltration of plaintext" - meaning decrypted content - "by mauling HTML emails." And, he says, "A few thoughts," which then follow in these following tweets.

He says: "In a nutshell, if I intercept an encrypted email sent to you, I can modify that email into a new encrypted email that contains custom HTML. In many GUI email clients, this HTML can exfiltrate the plaintext to a remote server. Ouch," he concludes Tweet No. 2.

Then he continues: "It's an extremely cool attack and kind of a masterpiece in exploiting bad crypto, combined with a whole lot of sloppiness on the part of email client developers. The real news here," he writes, "is probably about S/MIME, which is actually used in corporate email settings. Attacking and modifying encrypted email stored on servers could actually happen, so this is a big deal. Plus the attack on S/MIME is straightforward because it's," he says, "(A) a dumb protocol; and (B) a simple protocol not filled with legacy cruft; and (C) it's built into email clients."

He says: "Dumb and simple and one vendor" - that is, your client, your email client vendor - "to blame. But," he says, "of course the attack also implicated the garbage fire that is the PGP ecosystem. So of course that's what everyone is talking about." He says: "Over on HN the 'it's not PGP, it's mail clients' dance has begun, so I guess we have to talk about that." He says: "When it comes to PGP, the quality expectations on the crypto are low because it was invented in the Precambrian era" - yes, 27 years ago, and Phil Zimmermann was way ahead of his time, but things have changed. PGP hasn't.

He says: "It was invented in the Precambrian era, so it doesn't do proper authentication except as an optional afterthought. So in summary, PGP clients are vulnerable because 17 years after a vulnerability was known, the mitigation was not made a default in GnuPG, and defense was instead 'left to PGP clients,' which also make a convenient scapegoat when it goes pear-shaped."

Okay. So what he's referring to 17 years ago is that we learned that it was necessary to not only encrypt, but to authenticate. And we've talked about that often on this podcast. It is not sufficient to encrypt. You also have to absolutely guarantee against modification. And that's what PGP is missing. And as a consequence, the mistake made by GUI email clients can occur.

Okay. So get this. This is just so toe-curlingly cool. Email wanted to be able to contain more than just hello, some text, and goodbye. You wanted to have attachments. You wanted to have images. You wanted to do more than just a simple plaintext thing. So that created MIME, M-I-M-E, which is an extension. First of all, MIME stands for Multipurpose Internet Mail Extensions. And it allows multipart email where you have well-defined boundaries marking the beginning and end of sections.

Okay. So get this. This "mauling," what a bad guy does is they create a new email where the first section is the beginning of an HTTP image tag. So they do an open bracket IMG, space, and then SRC, where you're going to specify the source of the image. And then they use HTTP://evildomain, right, because this is the domain that is going to receive the decrypted email. Then they do a forward slash and end that section, that is, they leave the image tag source URL open. Then, as the second part of this new multi-part email, they drop in the original encrypted email that they can't read. It's encrypted, strongly encrypted by PGP or S/MIME, whatever. And then they close that second section.

Now, the third section is simply a closing quote on the URL and the closing angle bracket for the image tag. So what this is, is they've essentially created a new piece of email which is nothing but an image tag with a prepended domain to receive the email, and the encrypted body of the email as the URL of this image tag. So this is sent to the client. The email client says, oh, in order to display this, I need to decrypt this middle part, which is encrypted with PGP, or S/MIME, whatever. It decrypts it, and now you have all in plaintext an image tag where the URL from the root of the URL is itself the decrypted plaintext message. The client will URL encode it. So, for example, a space character, which is not URL safe, will get turned into a %20, for example, and so forth. And it will then query evildomain.com for the contents of the image, that is, to display the image.

Leo: That is cool.

Steve: Isn't that cool, Leo?

Leo: Yeah, that's very cool.

Steve: Oh, gosh, yes. And so what happens is evildomain.com has a query coming in to it to have an image displayed by the user's client. And the URL is the decrypted email. Oh, it's just sublime and gorgeous.

Leo: I do think, though, that it's a stretch to blame PGP and S/MIME.

Steve: Oh, agreed. It's not - no.

Leo: That's bad behavior on the email client's part.

Steve: Correct.

Leo: Conflating these three parts.

Steve: Although had PGP been authenticating, it would have recognized that the envelope had changed and then refused to decrypt.

Leo: Right. Which it should have done, obviously.

Steve: Yeah, yeah.

Leo: But this is just bad behavior. In fact, the email client I use for the most part, Claws Mail, doesn't do this because it doesn't do HTML.

Steve: Correct. And that is the advice is you want to use a client that will not do HTML because then it will not look at this image tag and go, oh, I need to go find the image.

Leo: Oh, oh, I've got to render it, oh.

Steve: Exactly.

Leo: Yeah.

Steve: So to conclude, the EFF says, under what to do about this, they say, and I think they're correct in this: "We are in an uncertain state, where it is hard to promise the level of protection users can expect of PGP without giving a fast-changing and increasingly complex set of instructions and warnings. PGP usage," they write, "was always complicated and error prone. With this new vulnerability, it is currently impossible to give simple, reliable instructions on how to use it with modern email clients. It's also hard to tell people to move off using PGP in email permanently. There's no other email encryption tool that has the adoption levels, multiple implementations, and open standards support that would allow us to recommend it as a complete replacement for PGP."

They say: "(S/MIME, the leading alternative, suffers from the same problems and is more vulnerable to the attacks described in the paper.)" They conclude: "There are, however, other end-to-end secure messaging tools that provide similar levels of security: for instance, Signal. If you need to communicate securely during this period of uncertainty, we recommend you consider these alternatives."

So anyway, very cool, I mean, mostly just sublimely clever attack. The idea of subverting image rendering on a client handling encryption to induce it to decrypt on the fly and then get the decrypted text exfiltrated via the URL of a remote image is just so cool. So that's really why I wanted to share this was I was like, just, wow, you know, very clever.

Okay. And so yes, Leo, I think that the takeaway is, for people who want to use email - so the danger is, just to explain it, if there exists the ability of an attacker to obtain or to have obtained encrypted content, the danger is that now they will create a new piece of email and send it to someone whose HTML-enabled PGP or S/MIME or it doesn't really matter what encryption will attempt to render the image and in the process send the cleartext back out. So anyone can disable that, prevent that by using a client to view email that will not render HTML. In which case...

Leo: Problem is everybody, every participant in the email has to be using that client. So there is a weak link. If there are multiple recipients, I'm not sure...

Steve: True. Good point, yeah.

Leo: That's a weak link. And Matthew Green points this out.

Steve: Good point.

Leo: The attacker can perform eFail attacks if only one of the participants is vulnerable. Now, what is unclear to me, maybe you could explain it, is he implies that even if a sender is vulnerable - doesn't it have to be that somebody with the private key would be, you know, somebody who could decrypt the email would have to be vulnerable. Because otherwise, I mean, sender can't decrypt. It can only encrypt it; right?

Steve: Yeah, I agree. I don't see how the sender would be vulnerable.

Leo: It implies somehow - but I think really, if I'm understanding this, really only somebody with the ability to decrypt the email.

Steve: Correct, correct.

Leo: So that person...

Steve: Because you are leveraging the client's own decryption, yes.

Leo: Right, right. However, if multiple recipients are intended, and you've encrypted with multiple public keys, then I guess you could then use one of the recipients' vulnerable email. So everybody should just stop using HTML email, which I've said for years. One more reason to hate HTML email.

Steve: Oh, boy.

Leo: I'm sure PGP will be fixed quickly on this. I would expect. No? Let me go look at GPGTools because that's what - I don't use PGP, I use the Open GNU Privacy Guard. So that means there's - this is in constant development.

Steve: The problem is this - yeah. Assuming that Matthew's solution, which is to say to add authentication, the sender needs to append, needs to add authentication, and the client needs to verify. So it's difficult, I mean, that's why it hasn't happened yet, even after 17 years.

Leo: The GPGTools people say that the next version of the suite, 2018.2, will include mitigations against this vulnerability and will be released this week.

Steve: Good.

Leo: So this is the one I recommend anyway, that people use GPGTools. And I presume that means - this is for the Mac. But I presume that means that GPG is also being fixed. And of course the other fix is to not use your email client to read your mail. But again, all intended recipients would have to be doing this.

Steve: Let's see. Actually, don't use your...

Leo: Use the command line.

Steve: Oh, okay.

Leo: You see what I'm saying? So I'm going to get a blob, an encrypted blob in my email client, take it, and decrypt it off to the side. Don't give the email client access to the decrypted version.

Steve: Right, right. Or actually, don't give anything that renders HTML access. So, like, even if...

Leo: Right, right. So put it on the - you could do it in the command line. That would be safe. Or put it in the clipboard, and you can - in most cases GPG can decrypt the clipboard. Maybe that would be risky, too. Put it in the command line. Save it as a text file and decrypt it with a command line.

Steve: And keep it away from your browser because of course browsers were the first things to do HTML.

Leo: Right, right, right. Wow.

Steve: Yeah. So Rowhammer becomes Throwhammer. Clever renaming. I like that a lot. And this is from some researchers from universities in Amsterdam and Cyprus that have very cleverly, I would say "invented," a brand new remote means for launching Rowhammer attacks via network packets and network cards. I can just imagine them sitting around brainstorming new ways to pound on RAM. And they realized that, in one of those aha moments, very much like whoever it was who figured out this incredible hack for PGP or encrypted email and using HTML rendering and multipart messages, these guys realized that the latest and fastest network connections were employing a technique known as RDMA. And it's actually rather widespread in high-end networking. As you might imagine, it stands for Remote Direct Memory Access. It turns out that's what all cloud providers and high-bandwidth networking solutions use.

RDMA has a Wikipedia page, lots of resources on the 'Net. And I'm going to share from their paper because they couch this, I mean, they explain this nicely, and also explain what they did. They said - and I'm just jumping in the middle. Now they're talking about the history. They said: "However advanced the attacks have become" - that is, previous Rowhammer attacks - "and however worrying for the research community, these Rowhammer attacks never progressed beyond local privilege escalations or sandbox escapes." And that's of course true. That's what we've been talking about.

They write: "The attacker needs the ability to run code on the victim machine in order to flip bits in sensitive data. Hence, Rowhammer posed little threat from attackers without code execution on the victim machine." And that's of course how we were, to some degree, keeping from staying awake at night and not worrying too much about what was happening. They say: "In this paper we show that this is no longer true, and that attackers can flip bits only by sending network packets to a victim machine connected to RDMA-enabled networks commonly used in clouds and data centers."

They say: "Rowhammer allows attackers to flip a bit in one physical memory location by aggressively reading or writing other locations, i.e., hammering the memory. As bit flips occur at the physical level, they are beyond the control of the operating system and may well cross security domains." Of course we've covered this extensively on this podcast.

"A Rowhammer attack requires the ability to hammer memory sufficiently fast to trigger bit flips in the victim. Doing so is not always trivial as several levels of caches in the memory hierarchy often absorb most of the memory requests. To address this hurdle, attackers resort to accessing cache eviction buffers or using direct memory access (DMA) for hammering. But even with these techniques in place, triggering a bit flip still requires hundreds of thousands of memory accesses to specific DRAM locations within tens of milliseconds. As a result, the current assumption is that Rowhammer may only serve local privilege escalation, but not be used to launch attacks from over the network.

"In this paper we revisit this assumption. While it is true that millions of DRAM accesses per second is harder to accomplish from across the network than from code executed locally, today's networks are becoming very fast. Modern NICs [Network Interface Controllers, LAN adapters] are able to transfer large amounts of network traffic to remote memory. In our experimental setup, we observed bit flips when accessing memory 560,000 times in 64 milliseconds, which translates to 9 million accesses per second. Even regular 10-gig Ethernet cards can easily send 9 million packets per second to a remote host that ends up being stored in the host's memory."

They ask the question: "Might this be enough for an attacker to effect a Rowhammer attack from across the network? In the remainder of this paper, we demonstrate that this is the case, and that attackers can use these bit flips induced by network traffic to compromise a remote server application. To our knowledge, this is the first reported use of a Rowhammer attack over the network. Specifically, we managed to flip bits remotely using a commodity 10-gig network. We rely on the commonly deployed RDMA technology in clouds and data centers for reading" - and of course that's the ideal target for this - "for reading from remote DMA buffers quickly to cause Rowhammer corruptions outside these untrusted buffers. These corruptions allow us to compromise a remote server without relying on any software bug."

So again, you couldn't get a better example of attacks never get worse, they only ever get stronger. We have now Throwhammer, which uses the fact that we've got very high-speed connections to servers, and those servers being the ideal targets as victims, being the recipient of bit flips that can then be used to get up to mischief. Like, for example, we saw, if you could flip the bit on memory access permission tables, you then immediately give a benign process running on that machine, but without any code execution capabilities, suddenly it gets access to all of memory on the physical computer.

So what this necessitates downstream will be that the communications buffers used by the NICs can no longer be in main memory. They cannot. They're going to have to be sequestered into memory, maybe static memory, maybe on NIC memory. Something's going to have to be done. Or the DRAM is going to have to be hardened so that it is no longer subject to Rowhammer attack. So another beautiful piece of work, and something that was a local-only attack now becomes networkable. So beautiful work; and, again, another piece of really nice engineering.

Leo: Nice. Throwhammer is a good name, too.

Steve: It's great.

Leo: Of course that's very important, to have the right name.

Steve: Got to have a good name and a good website.

Leo: Yeah. I was looking at the email client I use on the Mac, MailMate. And he said I was notified February 10th by Matthew Green and company, and mitigated mid-March.

Steve: Nice. It's been out there for a while.

Leo: It's been out there. And so one of the problems with that list that EFF has published and Matthew published is that it's on older versions of the software named, including MailMate. It's on a December edition, and he had fixed it by March. So you should just...

Steve: So update your clients and see whether the clients have already mitigated it themselves.

Leo: He said: "At the time I didn't put any real information in the update notes, the release notes, because I didn't want to telegraph what was going on. But I can tell you now that that was a mitigation for eFail." So one hopes that a lot of these clients would have handled it by now. Otherwise just keep using Mutt. You know, it works. It's good. Harmless.

Steve: Yes.

Leo: Mostly. Steve Gibson is the guy in charge of this whole kettle of fish, and we're glad he's here. You can find more about Steve at GRC.com. That's his website where you'll also find SpinRite, the world's best hard drive recovery and maintenance utility, and all the free stuff, tons of which is on the site. You could spend days just browsing through the site. But among other things you'll find this podcast there, not only audio, but also transcriptions of every word, beautifully carved into stone by Elaine Farris.

Steve: Lovingly transcribed.

Leo: Lovingly transcribed. So go to GRC.com. The transcriptions mean you can also search there for anything in any of the previous 600-some episodes. Have you hit 666? TWiT did. Oh, we've got three more before the...

Steve: We're at 663, yeah.

Leo: ...Mark of the Beast. All right. Coming soon to a podcast near you. You can get audio and video of the show from our website, TWiT.tv/sn. You can also subscribe. Or one thing a lot of people like to do is watch live and chat about it in the chatroom. There's always a lot of great back talk going on, and it's a great way to get additional information about the subjects we cover.

Here's the deal. We do the show about 1:30 p.m. Pacific on Tuesdays, that's 4:30 Eastern, 20:30 UTC. The chatroom is at irc.twit.tv. The live stream is at TWiT.tv/live. So enjoy. That's a great way to consume it. But even if you do it that way, you do want to subscribe so you'll have every episode on your Security Now! bookshelf. And you can do that in any podcast program. There are some written in Electron. You know, Electron, you just like automatically have hundreds of megabytes of support files, just like automatically.

Steve: Wow, wow. Horrible, horrible.

Leo: A lot of people don't like the idea of having Electron apps because each one has its own copy of Chrome. It's like installing Chrome Plus. All right.

Steve: Okay, my friend.

Leo: Thank you, Steve. Have a great week.

Steve: Talk to you next time. Bye.