TLS Interception INsecurity

Description: This week, Leo and I discuss the delay in this month's Patch Tuesday (we may know why!). Our favorite adblocker embraces the last major browser. A university gets attacked by its own vending machines. PHP leaps into the future. We cover a slick high-end Linux hack, the rise of fileless malware, some good advice for tax time, that it's not only Android's pattern lock that's vulnerable to visual eavesdropping, and what happens when you store a huge pile of Samsung Note 7's in one place. We've got some fun miscellany; a must-not-miss science fiction TV series; and, finally, a look at the growing worrisome security implications of uncontrolled TLS interception.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-599.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-599-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Our Valentine's Day special. That's why I'm wearing red. Steve's valentine to you, he says, the best picture he's ever shown on any episode. If you're not watching video, we'll describe it. We'll also get you all the news. For instance, it's Patch Tuesday. Where's the patch? It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 599, recorded Valentine's Day, Tuesday, February 14th, 2017: TLS Interception INsecurity.

It's time for Security Now!, the show where we cover your security online. Steve Gibson is here of the GRC, Gibson Research Corporation. Steve, before we begin, we should mention a couple of milestones. This is Episode 599. Next week is our 600th episode.

Steve Gibson: Woohoo! I did also notice that TWiT was 601. So I'm catching up.

Leo: We're two ahead. But TWiT takes - actually, we used to take time off. I haven't done that in a long time. And you never take time off. So I don't know. This may be a neck-and-neck race for some time.

Steve: I'm going to hang in there.

Leo: However, I'm also happy to say, I was just taking a look at the numbers, and this is now, after TWiT, the highest rated show on the network. It's TWiT No. 1, Security Now! No. 2.

Steve: Cool.

Leo: So congratulations. This show has been growing like crazy over the last two years. Something like 30% every year. So, and I think that that's because, well, it's a great show, but also people are very interested in this topic now, aren't they.

Steve: Well, it's certainly one that just keeps on giving.

Leo: Yes.

Steve: We have a bunch of fun stuff. The title for today's podcast is "TLS Interception Insecurity." A fabulous paper was produced after a bunch of very interesting research into, it turns out, the surprising number, larger by an order of magnitude than these experts in the industry expected, in the number of HTTPS connections which are being intercepted. It wasn't known to be that big. And unfortunately they found just a huge number of problems with this. So all listeners to this podcast know that this has been a source of angst for me for a long time. For a long time my position was corporations should absolutely not be doing this. And then I kind of came around, saying, well, okay, but how can they monitor their traffic if they're not able to look inside the connections which are increasingly TLS tunneled, not plaintext. And then of course we've also discussed the worrisome problem with the AV software, which is now doing this to end-users, typically without their having any awareness of it. So this paper is wonderful, and we'll wrap up the podcast with that.

But we have, interestingly, a delayed Patch Tuesday. And we may know why. We were talking about this last week. Our favorite adblocker has, well, it's more than an adblocker. It's not fair to call it that. In fact, Gorhill gets annoyed when people call uBlock Origin an adblocker. But it has embraced the last major browser that was a long-term holdout. An unnamed university has gotten attacked by its own vending machines. PHP leads the future in upgrading its built-in crypto library. There's a very slick high-end Linux hack that I'm not going to promote, but I did want to bring it to the attention of our major Linux people because it's very cool. We have the rise of fileless malware. Some good advice for tax time, which gave me an opportunity to revise a recommendation I had recently made in error. An observation that it's not only Android's pattern lock that's vulnerable to casual visual eavesdropping. Also a cautionary note of what happens when you store a huge pile of Samsung Note 7's in one place.

Leo: Uh-oh.

Steve: A little bit - uh-huh.

Leo: Oh, boy.

Steve: A little bit of fun miscellany. A must-not-miss science fiction TV series we've discussed before. I failed to bring it up last week, well, because I hadn't started into Season 2. And, oh, my goodness. It's just amazing. And then we will wrap up talking about some surprising revelations about just how bad TLS interception has become, not only because of its prevalence; but also, as always, the devil is in the details, and the devil's been busy.

Leo: Oh, boy. That'd be a good title for a show sometime: "The Devil's Been Busy." He's been busy. Okay. You've got a good Picture of the Week. You want me to show that?

Steve: The best picture of all time.

Leo: Okay.

Steve: This is the funniest thing I have ever seen in my life.

Leo: I'm looking at it. Oh, I get it. Oh, my god. Shall we see if people get it just by looking at it?

Steve: Oh, yes. It's just too wonderful.

Leo: What's happening here? So it's a motor or a generator, looks like.

Steve: Yeah, we have a gas, it looks like a gasoline-powered generator that's sitting on an elevated grid. And then there's this green solid copper wire, I mean green insulated, but you can tell it's solid copper, that comes out and then wraps several times around a pole, like a long rod, which is shoved into a half-filled pail, a plastic pail of dirt.

Leo: Do you get it, Burke? Do you get what that is, Burke? Burke's shaking his head, yeah, I know what that is. Maybe it'll help you if, instead of calling it dirt, you call that a pail filled with "ground."

Steve: Uh-huh.

Leo: Oh, man. A grounding wire? I don't think so.

Steve: Oh, I just love this. Now, I mean, as with anything on the Internet, you're tempted to think, okay, this is just, you know, this is a setup. But it's just so adorable.

Leo: John says we do that with all our racks. We've got buckets of dirt all around.

Steve: That's it. You need to hook your ground wire to a stake pounded into the ground. And here's a half-filled plastic bucket of dirt. Oh.

Leo: This is hysterical.

Steve: It's just wonderful. Anyway, so whoever - I wish I could remember who tweeted this to me. I saw it yesterday, I said, okay.

Leo: So funny. So funny.

Steve: This is the best thing I've ever seen.

Leo: Oh, my god.

Steve: It needs no explanation. I tweeted it late last night when I was putting this all together. And I got a great bunch of response from people, a bunch of humorous responses from people who got it. And it's just, oh.

Leo: Well, this is a great little test to see if people understand electrical power. If they're geeks, I think a geek would get this; right? If they look at it, and they go, what, what do you mean, what's wrong with that?

Steve: Yeah. You've got to parse it a little bit visually and go, wait, wait, what's going on here? And then you sort of follow the green wire over to the rod that's stuck into a plastic pail and say, okay, well, this is not what we really had in mind when we were saying you need to ground the generator. Oh, wonderful.

Okay. So Microsoft announced today, Microsoft Security dated February 2017, their formal announcement on Valentine's Day, today, 2/14, Microsoft Research Security Team said: "Our top priority is to provide the best possible experience for customers in maintaining and protecting their systems. This month, we discovered a last-minute issue that could impact some customers..."

Leo: Showstopper. That's a showstopper.

Steve: Uh-huh, "...and was not resolved in time for our planned updates today."

Leo: Wow. Wow wow wow.

Steve: "After considering all options, we made the decision to delay this month's updates. We apologize for any inconvenience caused by this change to the existing plan." Now, we don't know, okay, Ars believes this is the SMB. I mean, they're stating, and so maybe they have some other channel reporting because I haven't seen...

Leo: That was the big bug we talked about last week.

Steve: Yes. This is the Server Message Block problem, which is a zero-day because Microsoft, after being given 90 days to patch the flaw, never responded to the security researcher, Laurent Gaffie, who told them about this more than three months ago. And so he went public with a proof-of-concept written in Python, very nicely written, actually. You need to essentially pretend to be an SMB - that is, a Server Message Block - server. So he wrote an SMB server in Python. And the idea is that, at the minimum, you can crash a Windows system. This is any 8.1 or 10, also Server 2012 and 2016. So you need to arrange to get one of those to reach out to an SMB server.

The problem is historically there have been many ways to get a browser to do that, and that's all it would take to bring your system down. And US-CERT has been concerned that because the SMB client runs in the kernel, that would mean, if this could be matured from a crash into a remote code execution, then it would be possible to run a remote attacker's code in the kernel, which is as bad as it gets. So we don't know this is why it's delayed. And I'm a little suspicious because this should not be a hard problem to solve, unless - and I never took the time to dig into the nature of this exploit. But if it's just something that's easily patchable, it should have been easily patched months ago because this is a bad problem. So maybe the delay is for some other reason. We were expecting this, however, to certainly be fixed on this Tuesday's, today's, update. And we don't have an update today. So we don't know.

Leo: That's why, yup. Yup.

Steve: It certainly follows. Safari on macOS gets uBlock Origin.

Leo: I was just talking about that this morning.

Steve: Yes, I heard.

Leo: I wasn't sure if it did.

Steve: Yup. It doesn't...

Leo: Oh, interesting.

Steve: And I also heard you mentioning that your Chromebook immediately, when it synchronized...

Leo: Oh, yeah, baby. Oh, yeah.

Steve: ...and got its updates, that came in. So for our listeners who don't know, we did a podcast, #523. So here we are at, what, 599. So that was more than a year ago. I think it was like maybe September of 2015? Something like that. Anyway, it's podcast #523, for those who don't know and want to go back and bring themselves fully up to speed. I was a holdout for NoScript, which is what I'd been using for years. But it was just becoming too big a problem to have scripting disabled by default. So the big change for me was dropping NoScript and switching to uBlock Origin. We have full coverage in 523.

But I'll just mention that it has now, for a long time, been supported by the Chromium-based browsers, meaning Chrome and Opera; by the Firefox-based browsers, so Firefox, Firefox on Android, SeaMonkey, Pale Moon, and anything else derived from Firefox. Firefox does support an extra feature in uBlock Origin which is not available in the Chromium browsers, just because they lack the required feature in the API, which is script tag filtering, which is, as Gorhill says, "is of great help to foil attempts by many websites to circumvent adblockers." So this prevents circumvention. There is a Debian apt-get build that someone named Sean Whitton has produced for Debian 9 and later and Ubuntu 16.04 and later, which makes it very easy to add that to Linux. It's also supported under Microsoft's Edge; until now has not been available for Safari on macOS. It is now. It's available for download on GitHub. You might want to give it a few weeks to settle and stabilize because it's very new.

But essentially Gorhill takes exception to us calling it an "adblocker." He says it is not an adblocker: "It is a wide-spectrum blocker which happens to be able to function as," as he puts it, "a mere adblocker," as well. The default behavior of uBlock Origin when newly installed is to block ads, trackers, and malware sites. And it strongly leverages the publicly curated lists - EasyList, EasyPrivacy, Peter Lowe's ad/tracking/malware servers, various lists of malware sites, and its own filter lists. And he deliberately does not take donations. But he says, if you feel compelled to support this, then support the teams that are doing this list building because that's a public social service, essentially, which his engine uses to drive its selections.

So anyway, it's great to see it come to Safari. And it's funny, I've been running for a while. I've just sort of not noticed that I didn't have any blocker on my iOS Safari until I hit a site a couple days ago that was just intolerably obnoxious with ads. And I thought, don't I have one blocker turned on? And it turns out for some reason I'd turned it off. Probably some site was saying, oh, please don't block us, and I thought, okay, fine. Anyway, I turned it back on because it was just like it was impossible to use this site. There were, like, bouncing windows jumping around in front of me. I said, okay, this is just ridiculous. So uBlock Origin is what we now recommend. And obviously, Leo, you have remained a fan of it.

Leo: I still use it, yeah.

Steve: Yeah.

Leo: Mixed feelings about using an adblocker, but at least on the air I don't think it's inappropriate for me to keep that on.

Steve: Well, and unfortunately, malvertising has become a thing.

Leo: Yeah, yeah.

Steve: And so it's almost a defensive deal. I mean, I'm a supporter of the Google Contributor system. And we reported that they were going to be taking it down and then bringing it back up again. And, I mean, I'm happy to do that. When I browse a site, and I see - I have mine set to show those little pastel circles kind of moving around. And I see a page covered with rectangles that look like that, I think, okay, good. I've given some money to this site. I'm visiting it. And that seems like a reasonable tradeoff. So it would be nice if that sort of thing...

Leo: Google stopped it, by the way. I don't know if you noticed.

Steve: Right.

Leo: The Google Contributor thing is down. But I think that that's preparatory to a bigger rollout, I would guess.

Steve: Yeah.

Leo: Got to do more; right?

Steve: Correct. We're not exactly sure how they're going to change it or what they're going to do. And they haven't said when. But they said we just want to let you know we're shutting it down. Any unused balance will be returned to you, and we'll be coming back in the future. So it seems like a strange thing to do, like okay, why go through this dead period? But I'm sure they have their...

Leo: I'll tell you why. Google.

Steve: Yeah.

Leo: It's weird.

Steve: Exactly. Exactly. So this is an interesting story. And it's just so apropos to everything we've been talking about recently. We don't know what university this happened to. But this appeared in a Verizon four-page brag sheet about their involvement in this. But it was really interesting. And it was written in a first-person narrative format that I'll just share as is, although I've edited it a little bit.

So it reads: "Senior members of my university's IT Security Team rotate weekly as on-call Incident Commanders" - as they're called at the university - "in the event that a

response is needed. This week was my turn; and as I sat at home, my phone lit up with a call from the help desk. They had been receiving an increasing number of complaints from students all across campus about slow or inaccessible network connectivity. As always seemed to happen, the help desk had written off earlier complaints, and it was well after 9:00 p.m. when I was finally pulled in.

"I joined the conference bridge and began triaging the information. Even with limited access, the help desk had found a number of concerns. The name servers, responsible for Domain Name Service lookups, were producing high-volume alerts and showed an abnormal number of subdomains related to seafood. As the servers..."

Leo: What?

Steve: Yeah, seafood subdomains, you know, sushi dot something dot something.

Leo: Okay, mm-hmm.

Steve: "As the servers struggled to keep up, legitimate lookups were being dropped" - so this was a classic bandwidth flood or server flood denial of service where the servers were overwhelmed with seafood subdomains, and so legitimate queries couldn't get their IPs resolved, so this was preventing access to the majority of the Internet. "While this explained the slow network issues, it raised much more concerning questions. From where were all these unusual DNS lookups coming? And why were there so many of them? Were students suddenly interested in seafood dinners? Seemed unlikely. Suspecting the worst, I," he writes, "put on a coffee pot and got to work.

"Now that I had a handle on the incident in general, I began collecting and examining network and firewall logs. The firewall analysis identified over 5,000 discrete systems making hundreds of DNS lookups" - that is, within their internal campus network. "Of these, nearly all systems were found to be living on the segment of the network dedicated to our IoT infrastructure." Okay, now, so there's a little bit of good news. The good news is they have a segment of their network dedicated to their IoT infrastructure, meaning they have a segmented network, which of course is what we've also been talking about on this podcast now for some months, is the need to give your IoT devices their own network segment so when this happens, you have some control. And also so that when they get taken over - not if, when - then they'll be blind to the more important portion of your network.

So he continues: "With a massive campus to monitor and manage, everything from walkway light bulbs to vending machines had been connected to the network for ease of management and improved efficiencies. While these IoT systems were supposed to be isolated from the rest of the network, it was clear that they were configured to use DNS servers in a different subnet." So that's interesting. So they were on their own segment, but DNS had not been isolated. So this little mistake in isolation allowed the problem on the IoT segment to essentially bring down the rest of the campus. So there's an interesting data point.

"Of the thousands of domains requested, only 15 distinct IP addresses were returned. Four of these IP addresses and close to 100 of the domains appeared in recent indicator lists for an emerging IoT botnet. So what they found was they'd been infected by an emerging IoT botnet. This botnet was known to spread from device to device by brute-

forcing default and weak passwords. Once the password was known, the malware had full control of the device and would check in with command infrastructure for updates and change the device's password, then locking us out of those 5,000 devices.

"This was a mess," he writes. "Short of replacing every soda machine and lamp post, I was at a loss for how to remediate the situation." And really, think about it. They have 5,000 devices that have been commandeered, taken over, and had their passwords replaced. So this is not a small, I mean, like, this is the kind of thing that is actually happening now. And, I mean, as he says, how do you remediate this? How do you fix this?

He continues: "We had known repeatable processes and procedures for replacing infrastructure and application services, but nothing for an IoT outbreak. Fortunately, a less drastic option existed than replacing all the IoT devices on campus. Analysis of previous malware samples had shown that the control password, used to issue commands to infected systems, also was used as the newly updated device password. These commands were typically received via HTTP and in many cases did not rely on SSL to encrypt the transmissions.

"If this was the case for our compromise, a full packet capture could be used to inspect the network traffic and identify the new devices" - plural, times 5,000 - "password. The plan was to intercept the cleartext password for a compromised IoT device over the wire and then use that information to perform a password change before the next malware update. If conducted properly and quickly, we could regain control of our IoT devices." At this point I'd be holding my breath and crossing everything, all my body parts, fingers and toes and everything else, that this might work.

"While we waited for the full packet capture solution, I instructed the network operations team to prepare to shut down all network access for our IoT segments once we had intercepted the malware password. Short lived as it was, the impact from severing all of our IoT devices from the Internet during that brief period of time was noticeable across the campus, and we were determined to never have a repeat incident.

With the packet device underway, it was only a matter of hours before we had a complete listing of new passwords assigned to newly compromised devices. With these passwords, one of our developers was able to write a script which allowed us to log in and update the password and remove the infection across all devices at once." Whereupon I'd be going, "Whew!" "The whole process took a matter of minutes, and I made a mental note to save that script for later, although I prayed that we would never need it again. Now that the incident had been contained, we looked towards ways to prevent it from happening again."

So, lessons learned: Don't keep all your eggs in one basket; create separate network zones for IoT systems. Air-gap them from other critical networks where possible. Don't allow direct ingress and egress connectivity to the Internet; don't forget the importance of an in-line proxy or content filtering system. So essentially, retrospectively, they're wishing that they had already had a device in place that was already filtering all of their IoT traffic. They ended up installing that in order to obtain visibility into this IoT segment that then allowed them to capture the password and perform the remediation and also to take it off the Internet so that they were able to fix all of the compromised devices in isolation.

And then also change default credentials on devices; use strong and unique passwords for device accounts and WiFi networks. Regularly monitor - and this is where most people fall down. Regularly monitor events and logs. Hunt for threats at endpoints, as well as

the network level. Scan for open remote access protocols to your network and disable commonly unused and unsecured features and services - that's of course broadly good advice - that aren't being required. And include IoT devices - and this is difficult - in IT asset inventory because there are going to be so many of them. This university had 5,000 of these things. They had all of their vending machines plugged into the Internet in order to support and improvement management.

So an interesting and, I thought, really fascinating story. And of course it applies on a much smaller degree to end-users, to listeners of this podcast because, as I've said, we really do need to go to the effort to isolate the IoT devices. Clearly, we're headed towards a world where we've got devices that, for their functionality, need to be on the Internet. You just don't want their compromise to be able to then compromise the rest of your network.

Leo: You're still recommending the Ubiquiti EdgeRouter X as a way to do that?

Steve: Yes.

Leo: Okay, good. Somebody called the radio show this weekend and asked, and I said, yeah, as far as I know, this is still a very good way to go.

Steve: Yes.

Leo: You have to know what you're doing. You get on the web interface, and you have to know how to set up the VLAN stuff. But at least it lets you do it, with a 50, $60 router.

Steve: Forty, $49.

Leo: Forty, yeah.

Steve: Yes. And it's four or five ports and true isolation, where you're able - so it is a router where, instead of being a router connected to a switch, it is a router where every one of those ports is a separate Ethernet interface that can be given its own network. So you can have 192.0.x on one port, 192.1.x on another port, 192.2.x on a third port, and those networks cannot see each other. They are isolated. So it's just - it's a fabulous and very cost-effective solution.

But as you say, Leo, at this point you really do need to sit down and figure out networking. There are a lot of how-tos on the network. And I'm sure that the drive toward IoT and the problem with its security is going to end up with people coming up with a drop-in solution. If I didn't already have my plate overfull, I'd have some online configuration stuff for the Ubiquiti router. But I'll just let - because other people can do that, too. Those things are being created on the Internet.

So PHP 7.2: It just got voted to add libsodium to the core standard library.

Leo: Woohoo. Wow.

Steve: Yes.

Leo: That's interesting.

Steve: Yes. Last week the voting phase closed on an RFC to add libsodium to PHP 7.2. The result was unanimous: 37 in favor, zero against. Now, there was a much more split vote. In looking at it visually, I couldn't visually see, it looked to me like it was about 50-50, that is, the question was whether to keep the existing API namespace. It's like /sodium/something, or with this change to bind it into the root namespace, and they decided to do that. That did win. So it'll be - and I don't remember. I remember seeing it, but I don't remember what it was.

So it won't be down a hierarchy, it'll be in the root namespace. And so it's going to be built in. Libmcrypt is what PHP has had since 2007, so for, what, nearly 10 years. And that left only OpenSSL as a viable option for PHP 5.x and 7.0 up to this point. So of course by comparison libsodium is, as we know, it's my favorite library. It's the one that I used functions from for SQRL. We're seeing it being adopted pervasively across the industry. It's a modern, state-of-the-art crypto library offering authenticated encryption, high-speed elliptic curve crypto, and all the features that you want. So unlike other crypto standards, which basically are a potluck of various crypto primitives, for example, like WebCrypto, libsodium is comprised of carefully selected algorithms implemented by true security experts to avoid side channel vulnerabilities to offer short keys with high security.

And as a toolkit there was effort made in its design to prevent people from doing the wrong things. There aren't, unlike OpenSSL, where there are about 25 different ways to do something, there's one way to do something in libsodium, and it's the right way. So don't make it up, just call the function, and it'll keep you from hurting yourself. So bravo for the PHP team for doing this. And we're not going to see it immediately. It's going to be later, like toward the end of this year. But 7.2, when available, will have it built in. And that's just a great step forward.

Okay. So this is not for the faint of heart, but I wanted to put it on the radar of our Linux users because I know we have a huge Linux following in the podcast. This is a script called takeover.sh. It's up on GitHub. I've got the link to it. It's GitHub.com/marcan, M-A-R-C-A-N, the guy who did this, /takeover.sh. He describes it as: takeover.sh is a shell script to completely take over a running Linux system remotely via SSH, without rebooting, allowing you to log into an in-memory rescue environment, unmount the original root filesystem, and do anything you want, all without rebooting. You can even replace one distro with another without touching a physical console.

And so he warns people. He says: "This is experimental. Do not use this script if you don't understand exactly how it works. Do not use this script on any system you care about. Do not use this script on any system you expect to be up. Do not run this script unless you can afford to get physical access to fix a botched takeover. If anything goes wrong, your system will most likely panic. That said, this script will not itself," he writes, "make any permanent changes to your existing root filesystem, assuming you run it from the temp file system. So as long as you can remotely reboot your box using an out-of-band mechanism, you should be okay.

"But don't blame me," he writes, "if it eats your dog. This script does not have any provisions for exiting out of the new environment back into something sane. You will have to reboot when you're done. If you get anything wrong, your machine won't boot. Tough luck. This is not a guide for newbies." He writes, "I'm deliberately not giving you commands you can copy and paste." There's no curl where you click the link on a web page. "If you can't figure out," he writes, "what to do exactly without handholding, this script is not for you."

But anyway, I thought it was cool. This would allow somebody who knows what they're doing to SSH into a system and essentially, on the fly, switch from the mounted boot file system into an in-RAM temporary file system, and so much so that you are then able to unmount your normal operating boot file system, for whatever reason you may have, and perform surgery at a much deeper level than you normally could and then, finally, reboot the system and, if you've done everything right, have it come up under its new configuration. Anyway, I just thought it was very cool. Scary, but for people who are seriously into Linux, I could see where there could be an application for it. So I wanted to put it on our high-end Linux users' radar.

Dan Goodin, writing for Ars Technica, covered a story that comes out of Kaspersky that I thought was interesting. And it indirectly affects us because we've been talking about, we've been seeing the Mirai botnet where just rebooting devices gets control back. And this sort of shows us an interesting trend as a consequence of the fact that there are more and more Internet-connected things that do not routinely reboot, like your refrigerator or the Coke machine or whatever.

So Dan writes: "Two years ago, researchers at Moscow-based Kaspersky Lab discovered their corporate network" - their corporate network, Kaspersky - "was infected with malware that was unlike anything they had ever seen. Virtually all the malware resided solely in the memory of the compromised computers" - and, by the way, we did cover this at the time, but since then we're seeing a trend - "a feat [on the side of this malware] that allowed the infection to remain undetected for six months or more. Kaspersky eventually unearthed evidence that Duqu 2.0, as the never-before-seen malware was dubbed, was derived from Stuxnet, the highly sophisticated computer worm reportedly created by the U.S. and Israel to sabotage Iran's nuclear program." Which of course we covered extensively at the time, to spin their centrifuges up to a speed that would cause them to self-destruct.

"Now, fileless malware is going mainstream, as financially motivated criminal hackers mimic their nation-sponsored counterparts. According to research Kaspersky Lab plans to publish Wednesday [tomorrow], networks belonging to at least 140 banks and other enterprises have been infected by malware that relies on the same in-memory design to remain nearly invisible." And think about it. We detect malware by looking for it on mass storage. This is deliberately not storing itself where it can be scanned. It's in RAM, and we're not good yet at finding that.

"Because infections are so hard to spot, the actual number is likely much higher. Another trait that makes the infections hard to detect is the use of legitimate and widely used system administrative and security tools - including PowerShell, Metasploit, and Mimikatz - to inject the malware into computer memory." And in fact turns out that it was actually PowerShell. I'm sure we'll cover this because I'll have more details about this next week after having a chance to take a look at what Kaspersky has provided.

"What's interesting here," writes Kaspersky's expert Kurt Baumgartner, "is that these attacks are ongoing globally against banks themselves. The banks have not been adequately prepared in many cases to deal with this." He went on to say that people

behind the attacks are "pushing money out of the banks from within the banks" by targeting computers that run ATMs, automatic teller machines.

"The 140 unnamed organizations that have been infected reside in 40 different countries, with the U.S., France, Ecuador, Kenya, and the U.K. being the top five most affected nations. The Kaspersky Lab researchers still don't know if a single group of individuals is behind the attacks, or if they're being carried out by competing hacker gangs. The use of the fileless malware and command-server domains that aren't associated with any whois data makes the already difficult task of attribution nearly impossible."

So in general we're seeing a rise in this kind of in-memory compromise. It may be in many cases that firmware may not be safely writable on these machines, or can't even contain the malware. It's like, you know, as we know, our IoT devices tend to be memory lean, only having enough firmware space to contain their own OS. So there just, you know, there may be no space for that. So the malware lives opportunistically in RAM, hoping to stay there for as long as it can.

And many of these things, these IoT devices are appliances that are rarely, if ever, power cycled. And of course servers typically remain up between their patch cycles, if they don't crash. So malware that is scanning and is able to infect on the fly is able to take up residence until the system is updated and then rebooted after a patch cycle. And as we know, some routers, whether big iron or consumer little blue box routers, they may be up for years at a time with something living in RAM. So it's interesting because, as we know, scanning the device's file systems will reveal nothing because this stuff is no longer in files. It's arriving over the connectedness of the device and just taking up residence. Interesting.

And here's where I wanted to correct myself. This was a really nice blog posting that brought to my attention that I had referred to AxCrypt recently when I meant AES Crypt as the recommended encryption utility. Remember I was talking about email last week and saying, if you really care about the security of something you're sending to someone, TNO. You need to use the same protocol that we use for storing data, our data, in the cloud, or PIE, as we've also called it, Pre-Internet Encryption, where you encrypt it before you let the Internet have it. So anyway, this nice blog posting was titled "Encrypt Your Tax Documents Before You Send Them."

"It's that time. We're all doing our taxes online now. Some of us need some help, and that usually ends up with digitally sending tax documents. Do not ever," they write, "under any circumstance, email a document that has your Social Security number, home address, phone number, bank information, et cetera, unencrypted. That's a good way to lose all of your money and even your identity."

He writes: "I'm going to use two free services to send my encrypted documents out: AES Crypt and Dropbox. Step 1: Install AES Crypt. Get AES Crypt installed on your computer. Go here and install your platform." And I'll follow up with details here in a second. "Step 2: Zip all of those files. You want to send one encrypted blob of data." And I'm happy to see him use the word "blob." I wonder if he's a listener of Security Now!. "Put all your files into a single folder and zip that up. On the Mac, select the file and go to File >> Compress 'Sensitive Docs.' Step 3: Encrypt. Follow the directions for your platform to encrypt your zip file. On the Mac, drag the file onto the dock icon and enter a strong password when prompted. I suggest," he writes, "using a secure password generator" - oh, I think he is a listener - "like the one over at GRC.com."

Leo: Well, there you go.

Steve: "You can also use LastPass or whatever you use to manage your passwords. Make a note of that password, though. You can keep a copy of the password in TextEdit or nvALT - the documents are already on your computer. Step 4: Drop it in Dropbox. Using Dropbox makes sharing files easy. Put the encrypted file into your Dropbox folder and click Share Link. You can send them the file directly through the Dropbox website or get the link and send it separately.

"Step 5: Send the Password via a Separate Method." Yes. "If you send an email with a link to your encrypted document, it doesn't make sense to put the password into that email. Additionally, you might not want to send the password from your account to that same account. Try to send the password over iMessage, Signal, or another instant messenger, or to a secondary email address the person has. You could also" - and this is what I would do - "make the password something you could read to them over the phone."

Now, this person suggests "uppercase J, lowercase z, six, five" and so forth. I would prefer a long numeric password, grouping the numbers in pairs, because it's very easy to say 35274256. That's acoustically unambiguous. You get a pair of digits per utterance. And so if you just take a long numbers-only password - and of course you can have LastPass do that for you. You just tell it that you want numbers only and it'll say, okay. And then put spaces, and that's easy then to read over the phone to the other person. And then Step 6, this blogger writes: "The Takedown. Once the recipient has confirmed that they have the documents and have decrypted them on their own machine, remove the encrypted documents from Dropbox. Sure, they're safe," he writes, "but there's no reason to keep them online."

So about AES Crypt. I recently referred to AxCrypt, actually just last week, in this context, when I meant to say AES Crypt. We had spoken about it before, and I just got my tongue tied. AxCrypt was what I used to recommend, but it has gone commercial. And against all reason, they have a subscription plan, a few dollars a month or more per year. It makes no sense to me at all that you'd have a subscription plan for an encryption utility.

So bye-bye, AxCrypt. Hello, AES Crypt. The website is AEScrypt.com. It is free, open source, multiplatform - available on Windows, Mac, Linux, iOS, Android, and also for PHP and Java. So that's what you want to use. It is drop-dead simple. And as we know, AES is a good strong cipher. So you simply give it a good key. It'll use that key to encrypt your document. And then send it any way you want to. You could certainly do this.

And I presume this blogger is suggesting to use Dropbox because it can handle large things easier than email. But if it's not too large, then you could also just mail it to someone. But again, you do want to arrange to pass the shared secret key through some secure means. And again, I like making it very long, numeric only, chop them up into digit pairs, and then just dictate it over the phone, which is very easy to do.

Leo: This post kind of makes me sad, I'll be honest with you. PGP does all of this, and it does it better, and it does it right, and there's PGP key exchange servers that work very well and very reliably. It's public key crypto, not symmetric crypto, which you're using here. I just - and nothing wrong with this solution. But here's an

opportunity to get more people to use PGP or GPG, which is in the long run a far better solution. My opinion. Just throwing that in.

AES works fine. But why not use GPG? You don't have to worry about the symmetric key exchange because you have public key crypto. The problem is, of course, it's difficult to set up. But that's my issue, is I think fragmenting the market is not the right solution. The right solution is get everybody using PGP. People at Keybase are doing a great job making PGP very accessible, very easy. They have a key server. And I would say it's arguably more secure because you don't have the symmetric key exchange involved.

Steve: They are abandoning it, though.

Leo: Who's abandoning what?

Steve: Keybase is abandoning PGP.

Leo: No, that's not true. They're doing a chat - well, you know more about this than I do, probably. And I'm using their encrypted chat. But that's not using a GPG key. But you absolutely use your GPG key there. Keybase is still a GPG or a PGP key server. And that's the main purpose of it. They're not abandoning it. They're just not using it in their chat solution.

Steve: Well, they're abandoning it in the future. They are.

Leo: Are they? Is that what they said?

Steve: Yes.

Leo: Well, that's disappointing.

Steve: Well, I don't think PGP ever worked. I mean, you and I differ on this. I just think the hurdle is too big, and we need to reduce the hurdle.

Leo: Well, is there another better way to do public key?

Steve: It's on its way, yeah. Libsodium is all public key.

Leo: Right. And somebody's pointing out in the chatroom that you probably should be working with a CPA that supports HTTPS file uploads in the first place.

Steve: Very good point.

Leo: If your CPA is asking you to email your tax records, that's probably not the CPA to use.

Steve: Well, and of course I like this more as a generic solution because it's not just your CPA. It's anybody you want to send something to.

Leo: It's good for people to know about AES Crypt. Yeah, I agree. But I don't think a symmetric key solution is - I think that adds some fundamental problems. I think, we've got public key crypto, let's make it easier to use. Let's figure out a way to make it easier to use.

Steve: Yeah, yeah.

Leo: Just my two cents. You're the expert. I'm just throwing that in.

Steve: No, I don't disagree. So Sam Cox tweeted me an observation that just hadn't occurred to me when we were talking about the Android pattern lock. He said: "Numeric passcodes are also entered on a 3x3 grid." I guess technically 3x4. So, he said, "So aren't they also vulnerable to the Android pattern lock attack?" And it's like, yes, come to think of it, they are. Because the whole point of that was that you just - you have limited spatial resolution on a large grid. And what you're looking at is relative motion. And so the pattern lock attack exactly attacks a low-resolution keypad grid in exactly the same way. So bravo, Sam, for that observation. I thought that was great.

And Leo, oh, the second picture that we have here in the show notes, and it's just sort of sad. Snarky as always, theregister.co.uk called this a "Touching Note 7 Tribute." Which was the Samsung battery factory, which burst, spontaneously burst into flame.

Leo: Oh, lord.

Steve: Now, the good news is nobody was hurt. And my takeaway was just sort of shaking my head. It was: When you are recalling a huge number of batteries because they are prone to spontaneous combustion, it would be best not to store them in what amounts to a large - and this is my term - Fukushima pile.

Leo: Yeah. This is actually where they're recycling the Note 7 batteries; right?

Steve: Yes, right, right. So they're, like, all being returned, and this was the recycling facility. And unfortunately, I mean, maybe they just couldn't get to them all? I mean, clearly, I did a lot of travel, when was it, I guess over the holidays.

Leo: Oh, yeah.

Steve: And every, without fail, every single gate that you're at, the attendants are reminding us, now, if you have a Note 7, you need to just not bring it on the plane because you can't do that. But think about it, Leo. I mean, think about how many of them there were, that none of them are trusted, yet the battery has to be taken apart. I mean, it needs to be - the device needs to be opened, and the battery needs to be unwrapped because essentially it was wrapped over itself, and the corners were bent, and it was just manufacturing problems. But I never really thought about the problem of disposing of these.

Leo: Well, we had talked about this on The New Screen Savers with Kyle Wiens of iFixit. He says he doesn't know of a recycling factory that hasn't had a lithium-ion battery fire.

Steve: Wow. Wow.

Leo: This is very common because, if you puncture a lithium-ion battery in air...

Steve: If you breach it, yup.

Leo: ...it will burst into flame. And he says this is the real problem with these batteries that everybody's using now that are not in a protective shield. They're very hard to recycle. And he said everybody has fires. This is universal now.

Steve: Yeah.

Leo: So I guess you're prepared about that; right?

Steve: Yeah.

Leo: On we go. You've had your coffee. You've been caffeinated.

Steve: Recaffeinated.

Leo: Recaffeinated.

Steve: As opposed to decaffeinated. So a listener of ours shot me a tweet that I appreciated. He said: "pfSense allows one to activate UPnP and allow only user-specified IP addresses..."

Leo: Oh, I like that, oh.

Steve: Yes, "...on the LAN to open ports."

Leo: pfSense FTW.

Steve: Yup. And so that's a nice feature of pfSense. And I logged into my pfSense router, because that's what I'm using; and right there on the UPnP page, the second section says "UPnP Access Control Lists." And it reads: "These entries control access to the UPnP service. Client systems may be granted or denied access based on several criteria." And so you then have a standard router-style or firewall-style access control list where you're able to specify which devices can or cannot use UPnP.

So this is, again, for a higher end network environment, this is what you want, where you would assign a static IP, for example, to your Xbox based on its MAC address. That way it's always going to get the same IP because it's always going to - its interface, its Ethernet interface will always have the same MAC address. So you bind those together. And then you give it alone access to Universal Plug and Play so it can handle, it has permission to handle its own mapping, but nothing else in your network is able to see that. And alternatively you could use the network segmentation we were talking about because you're able to give access to a range of ports. So, for example, you could make your IoT segment able to use Universal Plug and Play, I mean, if you wanted to. Yet the other devices on your network could not. So a lot of flexibility there.

And there is, I meant to mention a couple months ago, there is a very nice little pfSense box. It's red, and it only has two ports. I wish it had more ports because you'd like to also be able to do network segmentation, although it does support VLANs. So if you had a VLAN-capable switch, which they're inexpensive, then you could have the little cute red two-port pfSense as your router and firewall, and then connect that to a VLAN-aware, inexpensive switch in order to then get multiple port segmentation in order to create a more powerful network. Anyway, we've got so many toys now to play with in our home networks, and none of them are very expensive. So anyway, thank you for reminding me that pfSense supports ACLs.

Okay, now, I have the Awful Pun of the Week, Leo. It's just...

Leo: All puns are awful, Steve.

Steve: Yes. I need to apologize profusely in advance. Now, in my defense, I grew up with a grandfather who lived to 103, who was the master punster of the world. And I've mentioned some of his favorites in the past. I think probably my all-time is "I opened the window, and influenza."

Leo: Yeah. On Valentine's Day, the old Marx Brothers pun: "When love comes in the door" - no, let's see. Oh, shoot. "When I come in the door, love goes innuendo." Something like that.

Steve: And he also, my grandfather, used to say, "If the rain keeps up, it won't come down." So anyway, with that understanding, many people saw this and sent it to me. And I don't know where it originated. But so this one is - and of course it's perfect for the podcast: "You can't use 'beefstew' as a password."

Leo: What?

Steve: "It isn't stroganoff."

Leo: That's good. That is a good - no, that's good. For a pun, that ain't bad.

Steve: You can't use "beefstew" as a password.

Leo: Why not, Steve?

Steve: It isn't stroganoff.

Leo: Yeah, that transcends pun into wordplay, as Logan5 says, yeah.

Steve: And I do like wordplay. Now for our next one. We have metaphysics and Schrodinger's cat meets cryptography. Now, this is a little along the lines of the sound of one hand clapping, or if a tree falls in the woods. So here it is, metaphysics: If you encrypt data with a key that is so strong that it would take more than all of the energy in the universe to crack, and the key is then destroyed, does the data continue to exist?

Leo: Oh, that's the old black hole conundrum.

Steve: Isn't that good? I like that, yeah.

Leo: That actually is kind of deep. That's what Stephen Hawking was debating, whether information can exist inside a black hole.

Steve: Right. And so in this case we have data which is encrypted using a key that would take more than all of the energy in the universe to decrypt. Then the key is destroyed. The question is, so does the data still exist?

Leo: Mmm.

Steve: Don't know.

Leo: So Groucho's quote - and I stand corrected. One of our chatters gave it to me. He said - oh, crap, I just lost it. Madame Swempski says: "I don't like this innuendo." Groucho says: "That's what I always say. Love flies out the door when money comes innuendo." I like mine better. Go ahead. Sorry.

Steve: Okay. And, finally, another follower and frequent tweeter friend of mine on Twitter said of the second season of "The Expanse": "Very realistic space combat. Makes 'Star Trek' look very silly and childish." And I was thinking exactly the same thing. I didn't mention it last Tuesday because I hadn't yet started to watch the second season. I was two episodes behind, or maybe one, and then the other one came in on Wednesday. Oh, my god, Leo. What I found myself thinking as I was watching this is I have only ever seen this in my mind when I am reading the best military sci-fi. There was some combat stuff in the first couple hours of the second season of "The Expanse" that is among the best I have ever - I've never seen it on the screen. I've only visualized it from reading it.

So, and we talked about this last year. "The Expanse" is a stunning Syfy series. It is now - the first season, for those who didn't already see the first season, is now on Amazon Prime. So if you are an Amazon Prime subscriber, you can catch up with the first season, which was excellent. It is just - it's the best science fiction on television without question. And the second - and I remembered that we knew this was coming. I read all four of the first books because the books are always better. But, boy, this move to the screen does not disappoint. So I just wanted to say to anybody else who thought, holy wow, I mean, this is just amazing television, yes, I completely agree. And for anybody who's not watching it, first season on Amazon Prime, and you're only two weeks into the second season. Definitely worthwhile.

And I did get a nice note from David Goldenberg. He said: "Hi, Steve. I've been a happy owner of SpinRite for a few years now. It's my secret weapon in the technology trenches. I am the family tech guy, I help out at my kids' school, and I have my own part-time business fixing PCs, training, and networking." So, yes, he's in the crossfire. He says: "SpinRite is always within reach, and never lets me down.

"Last week I'd been preparing a laptop for a presentation for my ARES Amateur Radio group. I volunteered to get a new program running to send text messages and email-type communications over the radio. After several days, I had everything working great, and spent several hours getting screenshots for the PowerPoint I was to prepare. I was getting together with another ham to go over what I had and to get his machine working.

"So I started up my laptop and got the dreaded BSOD" - which of course we know is Blue Screen of Death - "and an unmountable boot volume. I didn't break a sweat, or even worry, as I knew from experience that SpinRite would save the day. And needless to say, two hours later the drive scanned, several sectors were repaired, the laptop booted perfectly, and everything I needed was ready to go. You're great. Thanks! David Goldenberg, KJ6MCQ."

Leo: Nice.

Steve: And David, you're great for sharing your experience. I really appreciate that.

Leo: I like it.

Steve: And a ham, yes. A fellow ham.

Leo: Are you a ham? I keep - are you a ham?

Steve: I'm not.

Leo: You're not.

Steve: Never got into it.

Leo: Too analog for you, huh?

Steve: Well, yes, yeah.

Leo: Well, you do some analog stuff. Your light pen was analog.

Steve: Yes, indeed it was.

Leo: Early days, yeah.

Steve: Yup, yup.

Leo: But nowadays ham has become as much digital as analog, really.

Steve: Which I find really interesting, yes, the fact...

Leo: Me, too, yeah.

Steve: Yeah. Okay. So...

Leo: Let's go. TLS.

Steve: A fabulous piece of work. This was a big team with members from the University of Michigan, University of Illinois, Mozilla, Cloudflare, Google, UC Berkeley, and the International Computer Science Institute. And when I just scan their names, there are

lots of people that I recognize there. This was a beautifully assembled paper. And I started to go through it, trying to sort of like pull excerpts from it. And I just couldn't stop finding stuff that was all relevant. So I'm going to share the abstract and the introduction, both of which are just perfect.

The abstract reads: "As HTTPS deployment grows, corporate middlebox [as they term] and anti" - and so corporate middleboxes are these things, these appliances which corporations are using to intercept their Intranet's HTTPS communications to the outside. "And antivirus products," they write, "are increasingly intercepting TLS connections to retain visibility into network traffic. In this work, we present a comprehensive study on the prevalence and impact of HTTPS interception. First, we show that web servers can detect interception by identifying a mismatch between the HTTP user-agent header and TLS client behavior." And I inserted here my own observation: In other words, current TLS interceptors are not bothering to mask their presence, though they certainly could. Okay. So then they continue.

"We characterize the TLS handshakes of major browsers and popular interception products, which we use to build a set of heuristics to detect interception and identify the responsible product. We deploy heuristics at three large network providers: the Mozilla Firefox update servers, a set of popular ecommerce sites, and the Cloudflare content distribution network. We find more than an order of magnitude more interception than previously estimated and with dramatic impact on connection security.

"To understand why security suffers, we investigate popular middleboxes and client-side security software, finding that nearly all reduce connection security, and many introduce severe vulnerabilities. Drawing on our measurements, we conclude with a discussion on recent proposals to safely monitor HTTPS and recommendations for the security community." That's the abstract. So I'll just now share just the introduction because essentially this wraps up all of the meat of it. And then the rest of the paper, I think it was 14 pages, is just details. And actually it's all stuff that we've talked about in various contexts previously.

So they write: "When it comes to HTTPS, the security community is working at cross purposes." And we know this is true. "On the one hand," they write, "we are striving to harden and ubiquitously deploy HTTPS in order to provide strong end-to-end connection security. At the same time, middlebox and antivirus products increasingly intercept (i.e., terminate and re-initiate) HTTPS connections in an attempt to detect and block malicious content that uses the protocol to avoid inspection. Previous work has found that some specific HTTPS interception products dramatically reduce connection security; however, the broader security impact of such interception remains unclear. In this paper, we conduct the first comprehensive study of HTTPS interception in the wild, quantifying both its prevalence in traffic to major services and its effects on real-world security.

"We begin by introducing a novel technique for passively detecting HTTPS interception based on handshake characteristics. HTTPS interception products typically function as transparent proxies. They terminate the browser's TLS connection themselves, inspect the HTTP plaintext, and relay the HTTP data over a new TLS connection to the destination server. We show that web servers can detect such interception by identifying a mismatch between the HTTP user-agent header and the behavior of the TLS client. TLS implementations display varied support (and preference order) for cipher suites, extensions, elliptic curves, compression methods, and signature algorithms." In other words, sort of the metadata of the TLS connection varies from implementation to implementation that allows the target servers that are instrumented to detect those signatures.

So they say: "We characterize these variations for major browsers and popular interception products in order to construct heuristics for detecting interception and identifying the responsible product. Next, we assess the prevalence and impact of HTTPS interception by applying our heuristics to nearly eight billion connection handshakes." So this was a comprehensive study, eight billion connections. "In order to avoid the bias inherent in any single network vantage point, we analyzed connections for one week at three major different Internet services: the Mozilla Firefox update servers, a set of popular ecommerce websites, and the Cloudflare content distribution network.

"These providers serve different types of content and different populations of users, and we find differing rates of interception: 4% of Firefox update connections, 6.2% of ecommerce connections, and 10.9% of U.S. Cloudflare connections were intercepted. While these rates vary by vantage point, all are more than an order of magnitude higher than previous estimates." So think about that. Nearly 11% of U.S. Cloudflare connections - it's surprising - were being intercepted.

"To quantify the real-world security impact of the observed interception, we establish a grading scale based on the TLS features advertised by each client. By applying the metric to unmodified browser handshakes and to the intercepted connections seen at each vantage point, we calculate the change in security for intercepted connections. While some older clients' proxies increased connection security," meaning if you had an older client that was inherently low security, the running it through a TLS transparent proxy might actually increase its security because, for example, the client might be using RC4 encryption and not offering anything stronger, whereas the proxy does. So the weak security connection would be kept local, and the public connection would be stronger.

So they say: "While some older clients' proxies increased connection security, these improvements were modest compared to the new vulnerabilities introduced: 97% of Firefox, 32% of ecommerce, and 54% of Cloudflare connections that were intercepted were made less secure." Again, 97% of the connections to the Firefox update servers, where you would like to have security because you're getting a new copy of Firefox that could be compromised. And more than half, 54% of the connections to Cloudflare, of which 11%, 10.9%, were intercepted, more than half of those had their security compromised by that interception.

"Alarmingly, not only did intercepted connections use weaker cryptographic algorithms, but between 10 and 40% advertised support for known broken ciphers that would allow an active man-in-the-middle attacker to later intercept, downgrade, and decrypt the connection data. A large number of these severely broken connections were due to network-based middleboxes rather than client-side" - which is to say typically AV - "security software: 62% of middlebox connections were less secure, and an astounding 58% had severe vulnerabilities enabling later interception.

"Finally, we attempt to understand why such a large number of intercepted connections are vulnerable by testing the security of a range of popular corporate middleboxes, antivirus products, and other software known to intercept TLS. The default settings for 11 of the 12 corporate middlebox appliances we evaluated expose connections to known attacks; and five of those 12 introduce severe new vulnerabilities, for example, incorrectly validated certificates. Similarly, 24 of the 26 client-side security AV products we tested reduce connection security, and two thirds introduce severe vulnerabilities." That's 24 of the 26 client-side AV products.

"In some cases, manufacturers attempted to customize libraries or reimplement TLS, introducing negligent vulnerabilities; in other cases, products shipped with libraries that were years out of date. Across the board, companies are struggling to correctly deploy

the base TLS protocol, let alone implement modern HTTPS security features. Our results indicate that HTTPS interception has become startlingly widespread, and that interception products as a class have a dramatically negative impact on connection security. We hope that shedding light on this state of affairs will motivate improvements to existing products, advance work on recent proposals for safely intercepting HTTPS, and prompt discussion on long-term solutions."

I should mention that they also had a responsible disclosure, where all of the vendors of these sweepingly insecure products were privately notified. The document, this 14-page PDF has charts and tables showing devices which presumably have since been remediated, hopefully. But this really raises an interesting dilemma. And it's one for which there is not a clear solution. I mean, as the paper starts out, and as we said, the industry is at odds with itself because we are moving from plaintext to secure tunnels, yet there is a legitimate need to protect from the ability for malicious content of all kinds to hide within those tunnels. Even, for example, if you're doing nothing but looking for credit card numbers and the exfiltration of sensitive corporate data, that you could argue any corporation has almost an obligation to do. But if all the connections are secure, it can't.

So they had a couple takeaways. One was, they said: "We need community consensus. There is little consensus within the security community on whether HTTPS interception is acceptable. Discussions over protocol features that facilitate safer interception have been met with great hostility within standards groups. These communities need to reach consensus on whether interception is appropriate in order to develop sustainable long-term solutions."

And then they also said: "Antivirus vendors should reconsider intercepting HTTPS. Antivirus software operates locally and already has access to the local filesystem, browser memory, and any content loaded over HTTPS. Given their history of both TLS misconfigurations and remote code execution vulnerabilities," you know, we've talked about that the AV has become, has started introducing a larger attack surface than the OS it's trying to protect. Then they continue: "We strongly encourage antivirus providers to reconsider whether intercepting HTTPS is responsible."

And so the point they're making here is that, instead of trying to get the data on the fly, wait till it arrives and put your hooks in there, where you do have visibility. And I would say, as we did, I think, just last week or the week before, get rid of third-party AV now, at least on the Windows platform, because Microsoft, just as they did with the firewall and the previous firewall vendors, they've done this with AV and the hopefully soon-to-be previous AV vendors. There's a built-in solution that solves this in a secure way from a company that is on the ball, except for a little delayed Patch Tuesday today, on Valentine's Day, is keeping things current and updating monthly. Just go with the one that's built in.

And so, finally, I conclude by reminding everyone that, so long as the intercepting proxy, whether it is middlebox appliance or AV, as long as it does not have the ability to on-the-fly synthesize certificates which are trusted by the global public trusted root store - and no certificate authority will willingly, or at least publicly, allow this or they would become untrusted. This interception can only be done with the knowledge of the user's browser client which has chosen to trust the public key installed into it by the proxy's vendor.

So an interesting possibility here would be the explicit notification to the user when a non-globally trusted certificate is being used for any connection. In other words, for the browser, which certainly knows whether the certificate that has been used to sign this connection is part of the global trust store or was added on afterwards, that would be

nice to display to the user. Not that a user in a corporation has any choice. But at least being informed that your connection is being intercepted, even if it's benign interception, I think that's what we should have.

And we can have it. That is an unspoofable, robust compromise that at least tells the user something between here and there cracked this connection open in order to look into it. Maybe you have no choice, if you're in a corporate Intranet. But being informed, I think, is the least we can do. And until then, as an industry, we have a conundrum because we want our data to be secure, but we also want it to be safe. So anyway, great, great piece of work, and great research. And, wow, nice to know that 11% of connections are today being cracked open. And most of them have their security damaged as a consequence. So it's not even benign.

Leo: Anything as an end-user you can do about that? Not really.

Steve: No, not really. Now, most AV products give you an option to turn that off. That's certainly something you would want to do. It defaults on because they're thinking that's a benefit. I would turn that off. I would just say, eh. But then, of course, I mean, if you're a person who wants a third-party AV, you want it to scan the stuff coming in over HTTPS.

Leo: Right.

Steve: So it's like, well, you don't want to turn that off, because you want the benefit of it being able to be looking into your secure connections on your behalf.

Leo: Right.

Steve: The problem is, 24 out of 26 of them do it badly, so that it actually hurts your security.

Leo: Wow.

Steve: Yeah.

Leo: Kind of amazing.

Steve: We're just in a tough time right now. We need to get past this somehow. And we will.

Leo: Well, yeah. And in the meantime, you're here.

Steve: That's right.

Leo: Thank goodness.

Steve: And we'll be back next week for Episode 600.

Leo: Wow. Wow wow wow wow wow. Ladies and gentlemen, we do this show every week for 600 darn weeks. That's an awful lot. And we plan to keep on doing it, as long as you keep on listening. You can catch it live every Tuesday at 1:30 Pacific, 4:30 Eastern time, 21:30 UTC. Join us in the chatroom at irc.twit.tv. But if you don't want to do it live - and by the way, if you want to watch live, you can watch on our website, TWiT.tv/live or on the YouTube Live we have now, YouTube.com/twit, or at Ustream, or Twitch, Twitch.tv/twit. There are a lot of places to watch. Actually, if you go to our website, you'll have a choice. You can also watch or listen on demand. Steve has audio at his website, GRC.com, as well as transcripts. It takes about, what, a day or two after the show for...

Steve: Yeah. I normally get it either - so today is Tuesday. Normally Thursday morning I find that Elaine will have mailed them, and I get them posted as quickly as I can.

Leo: Show notes are also at GRC.com, and that's helpful if you want to see the pictures Steve talks about, or get the links to the research he's talking about. Where do you keep those? All in the same place as the shows; right? It's all in the same place.

Steve: Yup, yup. GRC.com/sn.

Leo: Okay. We don't have that, but we do have audio and video of the show at TWiT.tv/sn. And actually the way most people do it is they find their favorite podcast application, and they subscribe. That way they get it each and every week. You don't want to miss an episode. Is there anybody listening right now, show of hands, who has listened to all 599 episodes? Besides you and me, Steve. Oh, I haven't even listened to all 599 episodes. I've missed more than a few on vacations. I bet you half the audience has listened to every show. And the rest are trying to catch up. Bill in Michigan raises his hand. Good job. Strengths raises his hand. Yeah, all of them, all of them. Wait. Jay is saying, all? Yeah, all, 599 episodes. Thank you for listening. We appreciate it. And we will see you next Tuesday on Security Now!.

Steve: Thanks, Leo.