Never a Dull Moment

Description: It's been another busy week. We look at Firefox's changing certificate policies, the danger of grabbing a second-hand domain, the Fortnite mess on Android, another patch-it-now Apache Struts RCE, a frightening jump in Mirai Botnet capability, an unpatched Windows zero-day privilege elevation, and malware with a tricky new C&C channel. We find that A/V companies are predictably unhappy with Chrome, Tavis has found more serious problems in Ghostscript, and there's been a breakthrough in contactless RSA key extraction. As if that weren't enough, we discuss a worrisome flaw that has always been present in OpenSSH, and problems with never-dying Hayes AT commands in Android devices.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-678.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-678-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here with a potpourri of security stories: why Ghostscript is dangerous; the return of the Mirai worm (it's really on the upswing); and why AT commands, the Hayes command set, could be a danger to your Android phone. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 678, recorded Tuesday, August 28th, 2018: Never a Dull Moment.

It's time for Security Now!, the show where we cover your security and privacy online with this cat right here, this man, the myth, the legend, Steve Gibson.

Steve Gibson: The myth. The myth, I'm always a little surprised by that, Leo.

Leo: The mythic. How about that?

Steve: The mythological. The mythological.

Leo: Not mythological, mythic. Which would mean as if you were a Greek god, but you're not.

Steve: That's not like penultimate; right? Because it's not like second best, it's like...

Leo: Mythic: Relating to or - ah, here it is. Exaggerated or idealized.

Steve: Now, see, exaggerated...

Leo: That's not quite right either.

Steve: Neither of those really work for me, no.

Leo: The marvelous Steve Gibson, @SGgrc on Twitter. He's the Gibson Research Corporation, the man behind the Security Now! program for the last 13 years, now going on 14.

Steve: Into. Now, this is podcast number two of year 14.

Leo: Nice.

Steve: Counting down to the last 999 podcast. It's funny, too, how many people are upset by that. It's like, well, I'm sorry, but I still have a little hair.

Leo: And you reserve the right to change your mind if you feel like you want to keep going.

Steve: We could switch into alphanumerics, I suppose, so yeah.

Leo: Yeah. We could do hexadecimal.

Steve: Oh, and there have been lots of suggestions about how to fix this. So thank you, everybody. We are final podcast of August, and no one thing stood out in the past week. Surprisingly, all of the major disasters which were revealed by Black Hat and DEF CON and USENIX have pretty much been covered over the last three weeks. We do have two more USENIX reports which are interesting, and one DEF CON deal.

Anyway, so I titled this "Never a Dull Moment" because that is certainly true and of course always is. So we're going to catch up on yet another busy week. We look at Firefox's changing certificate policies. The danger - this is something I had never thought of before, and I'll bet if we listen carefully, Leo, when we explain this, we could hear a collective gasp from our audience around the world - the danger of grabbing a second-hand domain.

We also have the Fortnite mess on Android. Another "patch it now" Apache Struts remote code execution vulnerability. A frightening jump in Mirai botnet capability. An unpatched Windows zero-day privilege elevation that just happened a couple of ours ago, and a whole bunch of people tweeted to sort of hope I had seen it and to get it into this podcast, and we have. A malware using a tricky new command-and-control channel we've never seen before. Antivirus companies predictably being unhappy with Chrome's

decision to start pushing against them being, remember, all that "injected into" fun we had last week.

Tavis, who doesn't even need a last name, has found more serious problems in Ghostscript. He hasn't missed a year for the last few, and so yet again. We've also got a breakthrough in contactless RSA key extraction from devices. A worrisome flaw that's always been present in OpenSSH that only just came to light after two decades. And believe it or not, AT, the Hayes AT commands, Leo, that you and I grew up listening to, the sound of modems mating - actually, they did sound like unhappy geese - they're still present, and they're surprisingly dangerous. So lots to talk about this week. And a fun Picture of the Week courtesy of a good friend of mine. So I think a lot to do this week.

Leo: I do enjoy this one. It's funny because it's true.

Steve: It's a little, yes, you transient biological creatures. Our Picture of the Week is fun. It's a four-frame cartoon. The first frame shows a boxy-looking robot staring at a screen. And of course instead of saying "I am not a robot," the screen reads "Prove you are not a human." And then there's a blank to be filled in. Then the next frame of the cartoon shows us a side angle where we can see the robot's sort of arms are typing, and it enters the phrase, "There are no more humans." And then the third frame screen shows "Correct." And then the fourth frame, both of them, the screen and the robot from side view, are laughing, ha-ha-ha-ha.

Leo: Oh, man.

Steve: Okay, yes. I just thought it was fun and a little apropos of, as I said, oh, you transient humans. You just think you're all everything, while we come up with new AI chips and features and just, you know, we're so clever. We'll see what happens.

Firefox has begun the expected, well, not really a rollback, the expected untrusting of Symantec's root certificate and all certificates chained up to Symantec's root, with some surprising findings. It's not yet out in the stable release, but the nightly release for Nightly 63 now is no longer trusting any sites whose certificates were signed by Symantec directly or by any of their partners, including GeoTrust, Thawte, and RapidSSL, all whom Symantec acquired over the years and as a consequence are also no longer being trusted.

In the beta which is early next month this will appear, and then we're all going to see it, those of us who are using Firefox, by the end of October. October 23rd it will enter the stable release. And of course, as we know, Google has been leading a little bit. They took the same actions with Chrome 70's nightly build at the end of last month, on July 20th. Then it'll be in beta of Chrome on September 13th, and then it will land in Chrome stable on October 16th.

So at that point - and I don't know where Microsoft is on this. I've not seen any reporting, and I haven't gone digging for it. But what's interesting is that even, I mean, we've been talking about this, what, six months? Oh, and of course we also talked about how DigiCert has acquired Symantec's certificate business, and I'll follow up a little bit on that in a second. But as a consequence of that acquisition, everybody's happy because DigiCert offered to recertify under their root any affected customers. Yet even so, people are just apparently asleep at the switch, and some significant people.

Mozilla on their tracking page has noted that, for example, the Sony PlayStation Store, the Navy Federal Credit Union's online banking page, First National Bank of Pennsylvania's online banking, Estonia's LHV Bank, Canadian telecom Freedom, a couple of French banks, and the First National Bank in South Africa. Oh, and Intel's Japanese website. So some of them are obscure, but still large. And the point is that within a month anyone going to those sites with the majority browser on the 'Net, which is now Chrome, and certainly very popular, or Firefox - and again, I don't know what Microsoft is doing, but presumably Microsoft will be in here somewhere - is going to start getting very worrisome notes. And as we know, if you go somewhere with an untrusted root, it's either difficult or impossible these days to get around that.

So I hope that these companies get off, I mean, it's not like it's hard to remint yourself a certificate which is trusted by DigiCert, which is where Symantec has been aiming everybody. And as we know, DigiCert is my chosen certificate authority. Years ago, when I was leaving, who was it, VeriSign that I was no longer happy with because it just seemed like lots of nonsense dealing with them, so I switched to DigiCert, I mean, after doing a lot of research and deciding they were the guys for me.

DigiCert acquired Symantec's certificate authority business. That act jumped DigiCert from the position as sixth largest certificate authority by market share into the number three slot. So before the deal - and I'm surprised that Comodo is as big as they are. I don't really understand that unless it's because of their free certificate offering maybe. But as we know, we're not a fan of them. They've had all kinds of mistakes in the past relative to issuing certs. I'm surprised that they're still trusted, frankly. But the top six rankings before this were Comodo, IdenTrust, Symantec, GoDaddy, GlobalSign, then DigiCert.

After this deal, interestingly, IdenTrust is number one. But remember that they're the people who cross-sign Let's Encrypt's free and automated domain validation certs. So it's not surprising that any instrumentation that looks to see who is the root, since Let's Encrypt only just recently finally got into all of the last of the root stores, it's not surprising that IdenTrust is showing high because they're the people who have been anchoring and are arguably still anchoring Let's Encrypt's free certs. So second is Comodo, and in third place by market share now is DigiCert, and then followed by in fourth place GoDaddy and fifth place GlobalSign.

Anyway, this is, as we know, and covered at the time, Symantec made a series of mistakes. And the CAB, the CA Browser Forum, is understanding, and the browser vendors are understanding, of mistakes being made. What matters is how they are remediated when pointed out. And Symantec unfortunately did not impress a lot of people. It turned out that the number I remember was 30,000 certificates through time did not have proper management, and it was felt that this is not something - because the entire browser trust system is built on the integrity of the CA and their management of the process. Basically Symantec had third parties who they were allowing to do irresponsible certificate issuance. And that's not okay.

So the good news is I'm sure that the industry, the rest of the CAs have taken note of this and certainly don't want the same thing to happen to them. So anyway, this has been planned for a long time. This month and next month it actually happens. And again, it's like you've got to be scratching your heads why anybody is still using certificates that are going to be distrusted a month from now. But I imagine as soon as that actually happens, that'll get fixed overnight.

And speaking of certificates - and this, Leo, is where I said I could almost imagine a collective gasp being heard from our listeners. There is a site for this, as all good problems these days have. Insecure.design is the site. And this was - I think it was another USENIX presentation. It was some research that was done, and I'll catch up to it

in my notes. But they called it "BygoneSSL." Or perhaps, I was thinking, it should be "Gone, but unfortunately not forgotten."

And here's the problem, which in all of our discussions of certificates had never really occurred to me. As we know, certificates when issued can have a life of up to three years. Some of them are shorter. Some can only be two, some can be three, based on how tight the management of them is based on what they are asserting to be true. But domains can be abandoned. And, once abandoned, they can be reacquired by new owners. But the certificate that was issued to a previous owner of a domain is not automatically rendered invalid because the certificate, as we know, is tied to the domain name. So anyone picking up a secondhand domain needs to be aware of and cognizant of the fact that there may be, for as many as three years, a previously valid issued certificate for that domain in someone else's hands.

Of course we have certificate transparency now, and we have revocation. Except that we've extensively covered the fact that revocation is completely broken. I mean, it doesn't work at all. And so we're basically trusting the fact that certificates eventually expire themselves because they have bound into them, contained in the envelope which the CA has signed, is a "not valid after" date. So that's really the only thing that keeps this system working is that eventually the certificates will die. But until then, yeah, you have OCSP and some CRLSet stuff. But that only affects extended validation certificates on Chrome. And, I mean, it's a broken system.

So the question is, so how bad a problem is this? The research was done by these guys. And their abstract from their paper says: "When purchasing a new domain name you would expect that you are the only one who can obtain a valid SSL certificate for it. However, that's not always the case. When the domain had a prior owner" - and they said in parens "(or owners)" - "even several years prior, they may still possess a valid SSL certificate for it, and there is very little you can do. Using," they said, "Certificate Transparency, we examined millions of domains and certificates and found thousands of examples where the previous owner for a domain still possessed a valid SSL certificate for the domain long after that domain had changed ownership."

I'm reading from the abstract still. They said: "We will review the results from our ongoing large-scale quantitative analysis over past and current domains and certificates. We'll explore the massive scale of the problem, what we can do about it, how you can protect yourself, and a proposed process change to make this less of a problem going forward. We end by introducing BygoneSSL, a new tool and dashboard" - and they've got the link to it on their site - "that shows an up-to-date view of affected domains and certificates by using publicly available DNS data and Certificate Transparency logs. BygoneSSL will demonstrate how widespread the issue is, let domain owners determine if they could be affected, and can be used to track the number of affected domains over time."

So here's the numbers at the moment. The researchers took a sample set of 3 million domains and 7.7 million certificates to find how many certificates predated the registration of a domain while still being valid after the registration had expired. 1.5 million entries fit these parameters, and 25% of them had not expired at the time of the investigation. Meaning that, at the moment, 25% of 1.5 million entries - so, what, about a third of a million? - were still valid at the time of the research, meaning there were duplicate certificates, certificates that had been issued for a domain that its new owner wanted to secure. And somebody, somewhere, still had a non-expired valid certificate for the same domain.

Which, again, is like, in all of our discussions of this, somehow the issue of domain ownership change had never really occurred to me. Yes, certificates should be revoked. But as we know, that system doesn't actually work, nice as it would be if it did. So as a

consequence, it is really essentially up to the new owners of the domain to figure out, like to be aware of the problem, to check for the presence of previous owners' still valid certificates and, I would argue, to proactively look at the validity of those certificates. Has it been revoked? Where are the revocation lists? Are they current? How will the browsers currently handle that certificate if they're shown it? So anyway, real interesting piece of research, and something that just, like, whoops, you know, slipped through the gaps.

And speaking of slipping through the gaps, this Fortnite mess. Okay. So first of all, last week we talked about the so-called "man-in-the-disk" attacks. Maybe it was last week or the week before. Anyway, in the last couple weeks. This was the vulnerability which had come to light where many apps were staging themselves in external storage, which many Android devices have. It's a shared resource. And, I mean it's that sharing of the resource that creates the problem because many apps have access, overlapping access to that shared resource, creating the so-called "man-in-the-disk" attacks. So it turns out that the extremely popular first-person - is it a first-person shooter, Leo? I've never looked at what it actually is. I just know it's...

Leo: Yeah. It's Battle Royale. You go in, and a hundred other people go in, and last man standing wins. Yeah, it's kind of a first-person shooter, special kind, yeah.

Steve: So Google takes a 30% cut of all revenue generated by apps downloaded through the Google Play Store. And as you have often observed on the TWiT Network, Leo, I've heard you talking about Fortnite. Actually it's about Fortnite itself is free, but they make money, like a ton of money...

Leo: Yeah.

Steve: ...selling other stuff in the game. So over on iOS, where Fortnite has a player base of it's estimated between 125 and 150 million users, I think that's maybe across both platforms, during the first 10 days of Season 5 the iOS release netted Epic Games, the publisher of Fortnite, $2 million per day. $2 million per day. So the total mobile revenue on iOS between its March 15th release and the end of July was $150 million. Since there's no other method of distributing apps in the Apple ecosystem other than the Apple Store, Epic Games had no choice but to hand over a piece of their action to Apple.

But not so on Android. Epic Games has elected to bypass Google and the Play Store and to teach how to and encourage people, because there's no other way to obtain this latest Fortnite, how to sideload Fortnite by downloading an APK from Fortnite's own website. So this of course requires instructing people to push past all the sideloading warnings present on Android devices. And the big worry among the security community is that this could have the effect of encouraging and normalizing the sideloading process and behavior and pave the way to additional Fortnite clone malware. And of course we've previously talked about how there were Fortnite cheat add-ons that have been malicious and have hurt people.

So it turns out that on top of this, that is, that Fortnite has decided to sideload in order to not give Google a piece of their substantial action on the Android platform because it's possible, Fortnite is one of the apps vulnerable to this man-in-the-disk attack that we've been talking about, which the researchers at Check Point recently made clear. Google, maybe being a little miffed at Fortnite for encouraging sideloading, did not give Epic a 90-day window to fix after it came to light that the Fortnite installer was vulnerable to hijacking through man-in-the-disk attacks.

In Google's posting, they wrote: "The Fortnite APK [com.epicgames.fortnite] is downloaded by the Fortnite Installer to external storage." And then in the show notes I just clipped out a chunk of directory listings showing all of the entries there. And then Google continues, saying: "Any app with the WRITE_EXTERNAL_STORAGE permission can substitute the APK immediately after the download is completed and the fingerprint is verified."

Google writes: "This is easily done using a FileObserver. The Fortnite Installer will proceed to install the substituted fake APK. On Samsung devices, the Fortnite Installer performs the APK install silently via a private Galaxy Apps API. This API checks that the APK being installed has the package name com.epicgames.fortnite, but does nothing else. Consequently, the fake APK with a matching package name can be silently installed. If the fake APK has a targetSdkVersion of 22 or lower, it will be granted" - that is, this potentially malicious fake APK - "will be granted all permissions it requests at install time. This vulnerability allows an app on the device to hijack the Fortnite Installer to instead install a fake APK with any permissions that would normally require user disclosure."

Okay. So in response, unhappily, but immediately, Epic released v2.1. And as I said, Epic is clearly unhappy with Google's short notice. They said, actually it was Tim Sweeney at Epic who tweeted: "We asked Google to hold the disclosure until the update was more widely installed."

Leo: You can see why that's not going to happen, though.

Steve: Uh-huh. Yes.

Leo: Because it's out there. It's happening.

Steve: Exactly. Exactly. So Tim said: "Google refused, creating an unnecessary risk for Android users in order to score cheap PR points" is what Tim at Epic tweeted.

Leo: Really?

Steve: Google defended their timing, saying that their own instrumentation had shown that most Fortnite users had already updated to v2.1. So anyway, much has been made of this. And a lot of coverage in the press has been unhappy with Epic's decision. I mean, it's difficult to be, when you consider the amount of money that Epic is talking about. On the other hand, they're only making the money because Google has created the platform.

Leo: And I think you're risking your customers' safety to make some more money.

Steve: You are, yeah.

Leo: Now, was there already an exploit in the wild? I mean, it seems to me that the reason Google had to go public, well, I don't know, I mean, is to tell people to stop sideloading.

Steve: Right.

Leo: Or maybe there were no exploits going on?

Steve: Well, this doesn't prevent sideloading. It was that Fortnite's loader was...

Leo: No, I understand. But, I mean, the reason they exposed it is because people were actively at risk right now; right?

Steve: Yes, yes, yes, exactly, yes.

Leo: But were there people taking advantage? My sense was that it had already happened, that there were some...

Steve: It was essentially a zero-day, yes.

Leo: So you've got to tell people.

Steve: Yes.

Leo: So they stop doing it until it's fixed.

Steve: Yes, yes, yes. And immediately get the word out.

Leo: I mean, imagine if Google didn't tell anybody, and people were getting infected right and left. Oh, well, we wanted responsible disclosure so that Epic could fix it. Well, that's not responsible. That's irresponsible, I think.

Steve: Right, right, right.

Leo: Epic's just cheesed because they did the wrong thing, and they got bit.

Steve: Well, and as we talked about when we were talking about this whole man-in-the-disk problem, Google does clearly state that, for example, apps should do what Epic 2.1 now does.

Leo: Now.

Steve: Yeah, exactly. Now does. But they've always said this, that if you're going to use external storage, make very sure that you're actually installing the APK or whatever it is; that you can trust what it is that is in that store. And Epic just wasn't. They were like, oh,

just download some more Fortnite stuff, and let's just suck it right in. And this really did open them to the hack. And of course the security community is upset because the problem is sideloading is dangerous because you are training people to push past all of the warnings that Google has saying, you know, we haven't been able to check this. We don't know what this is you're loading. You need to be really sure. And here Epic is widely distributing to tens of millions, if not more, users their app. Not only was it insecure, but they're encouraging the circumvention of the Google Play Store.

Leo: Yeah, yeah.

Steve: So be interesting to see how this plays out.

Leo: That's exactly why we thought it was a bad idea to do the sideload.

Steve: Right, right. So Apache Struts once again...

Leo: Oh, no.

Steve: ...in the dog house, yes. Well, first of all, let's recall that it has not yet been a year since Equifax's massive failure to patch a previously well-known, long-known bug in Apache Struts was leveraged to expose the personal details of its 147 million consumers at an eventual cost to them, that is, Equifax, of $600 million. So that was an expensive patch to skip.

Well, we're back again. As we know, Apache Struts is widely used by enterprises globally. Last year it was estimated to be in use by 65% of the U.S. Fortune 100 companies who use it - it's a Java-based framework - who use it to build their web applications. So today we have a new remote-code execution flaw which allows attackers to remotely commander web servers. I've got the two links in the notes. A group named Semmle, S-E-M-M-L-E, Security posted - this is on the 22nd, so last Wednesday.

They said: "Today the Apache Software Foundation announced a critical remote code execution vulnerability in Apache Struts," they say, "a popular open source framework for developing web applications in the Java programming language. Applications developed using Apache Struts are potentially vulnerable. The vulnerability" - and I had to do a double-take because here we are at a CVE of 2018-11776, so into the five digits of the common vulnerabilities and exploits database - "was identified and reported from the Semmle Security Research Team, which works to find and report security vulnerabilities in widely used open source software.

"Organizations and developers who use Struts are urgently advised" - please, everyone, heed this this time - "urgently advised to upgrade their Struts components immediately. Previous disclosures of similar critical vulnerabilities" - they're just saying sort of rhetorically - "have resulted in exploits being published within a day." Remember, that's what happened last time. Within a day there was proof of concept, and then scanning began within a week of this thing going public. So that started last Wednesday, putting critical infrastructure and customer data at risk.

So under mitigation, switching now to the Apache disclosure and mitigation: "This new remote code execution vulnerability affects all supported versions of Apache Struts 2. A patched version has been released today. Users of Struts 2.3 are strongly advised to

upgrade to 2.3.35; users of Struts 2.5 need to upgrade to 2.5.17. The vulnerability is located in the core of Apache Struts. All applications that use Struts are potentially vulnerable, even when no additional plugins have been enabled." So this doesn't require anything more than just the core Struts package.

They say: "Struts applications are often facing the public Internet" - yeah, web apps - "and in most situations an attacker does not require any privileges to a vulnerable Struts application to launch an attack against it. To make matters worse, it is very easy for an attacker to assess whether an application is vulnerable," i.e., scanners are going to appear. "And it is likely that dedicated scanning tools will be available soon. Such tools will enable a malicious actor to quickly and automatically identify vulnerable applications."

So they said: "Whether or not a Struts application is vulnerable to remote code execution largely depends on the exact configuration and architecture of the application." Then they referred people back to their disclosure. "For more details, please see the section 'Was I vulnerable?' below." And they said: "Note that even if an application is currently not vulnerable, an inadvertent change to a Struts configuration may render the application vulnerable in the future. You are therefore strongly advised to upgrade your Struts components, even if you believe your configuration not to be vulnerable now."

So anyway, if you know, if you are a Struts-using organization, large or small, and you have any application publicly facing the Internet, to absolutely go to 2.3.35 or 2.5.17. Oh, and there's no issue with backward compatibility. This is a security fix only, nothing more. So it just changes that. So you don't need to worry about anything else being broken. Yikes.

Leo, I'm sad.

Leo: What?

Steve: I just, I don't know, maybe you already know about this. But the budget for the next fiscal year of NIST...

Leo: Yes, I know about this. I'm sad, too. And my clock's really sad.

Steve: Yes. I just saw this. That's so...

Leo: Yeah, WWV's discontinuing in 2019.

Steve: And, you know, it's the oldest, one of the oldest radio stations in the U.S. It was established in 1920. So almost a hundred years. I mean, I know that sound so well. I mean, I guess it's aging us a little bit.

Leo: Yeah, but that's not - is that the digital NTP server, or just the radio station?

Steve: Oh, just the radio station. But I also have "atomic clocks," as they're called.

Leo: Will they not work anymore? Will they not set themselves?

Steve: No. They won't set themselves anymore.

Leo: Oh. Thanks, NIST.

Steve: Well, with any luck, this'll get turned around. Well, it's the proposed cuts for fiscal year 2019 of the NIST from the White House, which is looking to cut back on things. So sad. Anyway...

Leo: NIST stands for National Institutes of Standards and Technology.

Steve: And technology, yup.

Leo: They run the WWV, what is it, out of Fort Collins, Colorado, the time?

Steve: Yes, exactly. In fact, I kind of point my clocks in that direction because they have a long wave antenna.

Leo: Sometimes I just put them outside because, yeah, inside they don't always update.

Steve: Yeah, yeah. Okay. So the Mirai botnet that we have spoken of often has gained a worrisome new trick. It has started leveraging, or a version of it, the Sora, S-O-R-A, variant has started leveraging Aboriginal Linux, which has made it suddenly much more dangerous. Aboriginal's slogan is a little prophetic. It says: "We cross compile so you don't have to." This new Sora variant of Mirai is compiled with Aboriginal Linux, which offers a toolchain utility which allows its single base source to generate binaries for a large number of different platforms, many more than previous variants of Mirai could infect. So in other words, by writing Mirai to be compatible with Aboriginal's toolchain, they're suddenly able to produce a vast array of Mirai binaries.

So what's happened is, and it's been spotted by researchers, once Mirai, that is, the Sora version or variant of Mirai, accesses a device by guessing its SSH password, its infection routine now successively downloads and tries binaries from a long list of Sora binaries, one by one, until it finds the one which is appropriate for the infected device's platform. And as a consequence, Mirai has made a cross-species jump to Android and Debian, where Mirai has never before been present. And I have here at the bottom of this page of the show notes a chart showing the rise in Mirai infections from near zero in June - or is that July? I don't know how the dates are organized. But, I mean, it's just gone...

Leo: 2018-06-03, 06-17...

Steve: Yeah, so it looks like June 3rd to August 12th. So in a couple months.

Leo: Boy, it seems precipitous, wow.

Steve: Yes. It has really, really jumped.

Leo: What is the left scale, though? I'm always suspicious when I see graphs with no left-hand scale.

Steve: Yeah. And as you know, it's a pet peeve of mine when people do non-zero-based charts.

Leo: This looks like it might be zero based, it was so flat at the beginning.

Steve: It really does. Yes, it doesn't just show it like already starting up. So it does look like it's zero-based. So anyway, just a heads-up. We're not out of the woods yet with Mirai. And the bad guys are getting more and more clever. I mean, again, it's a broken record on this podcast. But we absolutely have to have autonomous devices that have an Internet face able to update themselves. It has to happen. And we know that some HP printers can now be configured to do so, thank goodness. Our routers have to do that, as well.

Okay, now, I can't read the following tweet in its entirety, or without abbreviating...

Leo: What is it, from the President?

Steve: In this case, no. It's SandboxEscaper. He tweeted a GitHub link. But the tweet reads: "Here is the ALPC" - that stands for Advanced Local Procedure Call - "ALPC bug as a zero day." And then the GitHub.com/sandboxescaper link. And he says: "I don't effing" - and he didn't abbreviate - "care about life anymore. Neither do I ever again want to submit to MSFT" - that's of course the abbreviation of Microsoft - "anyway. Eff all this stuff." And he didn't say "stuff." So that was his tweet, surprising the world with that tweet to a never-before-seen, since validated, local privilege elevation that is effective on fully patched 64-bit Windows 10 systems.

Shortly after this SandboxEscaper's tweet, CERT, the vulnerability analyst Will Dormann, whom we've quoted before, verified the authenticity of this zero-day bug, tweeting: "I've confirmed that this works well in a fully patched 64-bit Win10 system." He says: "LPE right to SYSTEM," all caps, meaning that any restricted process can immediately elevate itself to full system, that is to say, root or kernel privilege, using this today. And frankly, Leo, I was wondering whether it would still be up on GitHub because of course Microsoft now owns GitHub, and it's not good for them for this to be there. But it's there. And so props to them for not starting to edit GitHub for things that they like and don't like.

Anyway, according to the short advisory which has since been published by CERT, of course, this zero-day flaw, if exploited, could allow local users to obtain elevated system privileges. But of course, since local procedure calls are, as their name suggests, local, the impact of the vulnerability is at least limited so that it received a CVSS rating of between 6.4 and 6.8. But as we know, privilege elevation bugs are still highly sought after since they allow anything that gets into the machine to make much deeper changes, that is, to operate with full root system kernel-level privileges.

So SandboxEscaper did not notify Microsoft, which leaves Windows users vulnerable until a patch is released to address the issue. I mean, if this were a zero-day remote code execution against all fully patched Win10, Microsoft could be expected perhaps to release an emergency update, an out of cycle. I don't think they're going to. We've got the second Tuesday of the month. You'll be cruising some river somewhere.

Leo: It's called the Mediterranean Ocean, Steve.

Steve: Oh, you'll be on the Mediterranean while we're busily patching our Windows 10 systems.

Leo: I'm not actually bringing a Windows system with me, so I won't have to worry.

Steve: Well, when you get home, do not delay, yes. And we expect that second Tuesday of the month. This is probably not going to be difficult for Microsoft to fix. And so I will be very surprised if it's not bundled into what we get on September 11th. Ooh, September 11th.

Leo: Yeah, hmm.

Steve: So malware has been found leveraging a tricky new command-and-control channel discovered in the wild, actively exploited, affecting, interestingly, not only MS Outlook, but The Bat!. I don't know why they chose The Bat!, but that's like this...

Leo: I like The Bat!, actually.

Steve: I was going to say, it's a well-regarded, very complete email client. I looked at it at one point when I was considering leaving Eudora because it's like, hey, it looks like a good alternative. So this thing's called Turla, T-U-R-L-A, is the name given to it. It was first seen in 2013, so five years ago, though it's largely been quiet. In that code there are some timestamps dating back to 2009, but their authenticity is questionable. Maybe they were some other stuff that was just brought in in 2013, not recently compiled. ESET Security did a nice report with all the details about this Turla Outlook Backdoor, as they called it. And what prompted them was they found a much updated and far more advanced version which transacts with its masters, wherever they are - get this, Leo - using email as the comm channel and PDFs as the carrier.

Leo: Wow.

Steve: So, yeah. So the bad guys can email a PDF to a user who is known to have been infected with Turla. The actual backdoor is contained in a single self-registering DLL. So all something has to somehow arrange is to be able to execute this DLL, either use regsvr32 or it's got a self-registration capability. Get it to register. It then becomes persistent and lives in the system. When an Outlook or a Bat! user receives email, the backdoor is scanning their inbox, spots the PDF, grabs it and deletes it. And apparently it actually creates a little transient appearance in the inbox, which might cause someone to

do a double take. It's like, what? Wait. Wasn't there something just there? But then they just chalk it up to, you know, ghosts.

Anyway, the PDFs are weaponized that contain instrumentation. The ESET guys reverse-engineered the technology, found a whole list of commands that can be encoded in PDFs to allow them to do things to the infected user. So while the infection is present, all outgoing emails are forwarded to the attackers. So it's a full spy on anything that user sends. Metadata, so email addresses, subject, and attachment names of incoming emails, are logged.

And among the commands possible, the attackers can request that any file on the system be sent out via the backdoor. The backdoor can execute additional programs and/or commands; can download additional files, exfiltrate anything, receive commands, as I mentioned, via PDF email attachments; and, ESET said, is highly resistant and resilient against takedown. So an interesting and sort of unsuspected command-and-control channel. Basically your MS Outlook or your The Bat! client are compromised; and, when someone sends that person a compromised DLL, that gives them command and control. And then a PDF is packaged up and sent back out as by way of response. So it uses PDFs as the transaction mechanism and has access to anybody it manages to get infected. Wow.

Okay. We talked about last week, we had fun with the idea of Chrome not wanting to be injected into. And that coverage was brought to us by Bleeping Computer. Of course, and Bleeping Computer, as we know, they first got onto our radar because they were really the go-to website. Lawrence Abrams is the founder of Bleeping Computer. They were the go-to website for cryptomalware and were the first really good information about that. So they reached out to a bunch of AV companies with whom they maintain of course good relations because that's their business.

BitDefender's Bogdan, looks like Botezatu, B-O-T-E-Z-A-T-U, who is a senior e-threat analyst for BitDefender, told Bleeping Computer's reporters that as of last Monday, August 20, BitDefender would no longer be monitoring Chrome 66 and subsequent versions with their anti-exploit technology. Bogdan said: "Starting with the Chrome browser v66, Google has gradually rolled out a new feature that prevents third-party software from monitoring the application's processes." Well, we know that's not quite true. It's that it's beginning to notify users if the browser crashes and then has a list of things that might be culprits.

Anyway, he says: "While this measure ensures that rogue applications do not interfere with the Google product, it also prevents security solutions from inspecting the browser's memory in search of potentially dangerous exploit code. With v66, Google Chrome displays" - okay, now he's explaining it correctly - "a post-crash warning asking users to remove the security solution if it monitors the browser's processes, even if the security solution is not responsible for the respective crash." And, you know, frankly, as a developer, it's not clear to me how to ascribe responsibility. If the browser just crashes, Google has watcher processes that will notice that there was a crash. But it can't necessarily detect how, exactly, or who was responsible. It's crashed.

Anyway, he says: "In order to prevent this message from occurring and having users unwarily uninstall the security solution, which would leave them exposed to a vast array of online threats, BitDefender has issued an update to stop the Anti-Exploit technology from monitoring the Chrome browser." In other words, BitDefender doesn't want you to remove it completely. So they've just removed that one tentacle of BitDefender from Chrome so that you won't remove - so that their users won't be induced to remove all of BitDefender. "The update was delivered to customers on August 20th at 7:00 a.m. Eastern time."

He finishes: "As a leading global cybersecurity technology company, BitDefender is committed to providing cutting-edge, end-to-end cybersecurity solutions." This is just a commercial. "We regret being forced into removing protection for one of the world's most popular browsers, and we urge users to not uninstall their security solution they have installed on their computers." So they were probably either already or correctly concerned that this would cause people to remove BitDefender if Chrome popped up and said BitDefender may be causing Chrome to crash. I mean, why wouldn't you? So they're stopping that.

Kaspersky said: "Kaspersky Lab is aware of Google Chrome showing alerts that the company's applications are incompatible with the browser. We have contacted Google to find a solution, and we are continuing to look for possible workarounds to resolve this issue."

Leo: Yeah, but read the next paragraph. Read the next paragraph.

Steve: Yes. "Having our code injected into the Chrome browser is an important part of the overall Internet security approach implemented by security vendors to provide users with safe web surfing."

Leo: I rest my case. Oh, lord.

Steve: Yeah. "For example, it is critical for a feature of Secure Input" - their name - "that blocks attempts of stealing sensitive data like credit card number, login, password, with malware (keyloggers) installed on user's devices." Yeah.

And Malwarebytes, similar sort of thing. Avast AVG, they were also quoted: "We have fixed this issue, and our products are not reported by Chrome."

Leo: How do you fix it?

Steve: Well, you stop. Basically all these guys are abandoning their real-time in-browser monitoring, which up until now they've all been doing in order to protect their customers.

Leo: What do you think? I mean, is that a legitimate useful thing?

Steve: I don't know what Google is thinking. Google could offer documented hooks which would allow programs to do this. They could protect them with "are you sure" and "are you really, really sure." I mean, and so the idea being that the question is, or my assertion is, it could be done in a way which is compatible. Microsoft has done this. Microsoft - we went through the same process with Windows in the early days where AV stuff was getting its hooks in. It was modifying API calls and causing instability for Windows. Microsoft said no, but in return gave firewalls and AV sanctioned ways of filtering the same stuff.

And apparently, I mean, so also it's noted, remember that Chrome also said that Edge is not allowing, is intolerant of people sticking their hooks into Edge, not allowing injection into the Edge processes. So Google is arguing we're doing the same thing with Chrome. We're not wanting our browser to be crashed by other people's code which introduces

instability. I think that's valid. What I don't have a read on is whether this is a "me too" checkbox so that everybody is saying, oh, yes, we hook your browser and are able to rummage around in its memory, or do we already have that protection from outside of Chrome? I'm not on the inner side enough to know. But I do know that Chrome could make it possible and sanctioned and safe if they chose to. And apparently they're choosing not to at this point. Maybe that'll change.

And what'll happen is users will just lose that protection. They'll still keep their AV. Chrome won't crash. But there will be maybe additional vulnerability as a consequence. And in fact Lawrence, who I think did the reporting for the story, is not happy at the idea that AV is being kicked out of Chrome. His feeling is that's not a good thing for users.

Okay. And heard of Ghostscript? We all have, those of us who pay attention to what libraries get installed in our machines and what various things are using. It is the number one most popular and very widespread interpreter for Adobe's PDF and Postscript that is not Adobe's. And brought to us by none other than Tavis Ormandy of Google's Project Zero is the revelation of, and for which there is no current patch, a serious set of vulnerabilities in Ghostscript.

It is, as I said, it is the industry's number one open source interpreter for PS and PDF page description. It's embedded in hundreds of applications and used by code libraries which allow desktop software and web servers to handle Postscript and PDF-based documents. ImageMagick is one of the more popular apps which embeds Ghostscript. Evince and GIMP also do so. And pretty much any non-Adobe software which supports PDF creation, editing, viewing, or printing has Ghostscript in it somewhere. Exploiting the vulnerabilities which Tavis has enumerated allows an attacker to take over applications and servers that use Ghostscript. And I'll say again, no patch is available.

Tavis tweeted: "This is your annual reminder to disable all the Ghostscript coders in policy.xml." I should note policy.xml is a component of ImageMagick, and it allows you to control what the codecs, essentially, the image codecs which ImageMagick uses. So you are able to disable ImageMagick's use of Ghostscript, which is what Tavis is strongly recommending. And I note that Tavis's tweet reads as it does, that is, this is your annual reminder, because he's previously identified problems in 2016 and 2017. He posted to the Chromium bugs page under the title "Ghostscript: multiple critical vulnerabilities including remote code execution." Then he said: "I sent the following mail to the OSS security mailing list." And of course that's seclists.org. He said: "These are critical and trivial remote code execution bugs in things like ImageMagick, Evince, and GIMP, as well as most other PDF and PS tools."

And so his posting reads - and I'm going to skip most of it, but it starts: "Hello, this was discussed on the distros list, but it was suggested to move discussion to oss-security. You might recall I posted a bunch of sandbox escapes in Ghostscript a few years ago." Then he links to it. He says: "I found a few file disclosure, shell command execution, memory corruption, and type confusion bugs. There was also one that was found exploited in the wild." So this is under active use.

"There was also a similar widely exploited issue that could be exploited identically." Then he says: "TL;DR: I strongly suggest that distributions start disabling PS, EPS, PDF, and XPS coders in policy.xml by default." So the idea would be, if distributions did that, then things like ImageMagick would be safe for their users pending Ghostscript being fixed. But Tavis is not suggesting that. He thinks Ghostscript is beyond repair. And of course I use the "I" word. It is an interpreter. And it's a huge, massive, hairy interpreter, and we know what a problem interpreters are.

Anyway, so I skipped a whole bunch of stuff in his posting. And he says: "For example of what these exploits [he enumerates] do," he says, "This means you can create a file in

any directory." And he says: "I don't think you can prevent the random suffix," which refers to his exploit. He says: "Additionally, I have a trick to let you read and unlink any file you have permission to." And he says later: "This can be used to steal arbitrary files from web servers that use ImageMagick by encoding file contents into the image output." And he says: "See my previous proof of concept here for an example, i.e., you can convert malicious.jpg thumbnail.jpg, produce an image with the contents of a file visible" to the ImageMagick app, or the DLL which is where Ghostscript is located.

He says: "These bugs were found manually," meaning just by his inspection of the source. He says: "I also wrote a fuzzer, and I'm working on minimizing a very large number of test cases that I'm planning to report over the next few days." So more may be coming. He says: "I'll just file those issues upstream and not post each individual one here. You can monitor," he says, it's "https://bugs.ghostscript.com if you want to. I expect there to be several dozen unique bugs.

"In the meantime," he finishes, "I really *strongly* suggest that distribution start disabling PS, EPS, PDF, and XPS coders in policy.xml by default. I think this is the number one unexpected Ghostscript vector. In my humble opinion," he says, "this should happen ASAP." So anyway, he feels that the whole issue of it is a fragile security boundary that should not be tested. So I imagine we will be seeing Ghostscript updates. On the other hand, it being as popular as it is means that it is spread all over systems that use Ghostscript for PDF rendering applications and will be difficult to root out completely.

And under the category, quoting Bruce Schneier, of "attacks never get weaker, they only get stronger," a presentation at that recent USENIX conference two weeks ago in Baltimore detailed a new technique for retrieving the public key encryption keys, meaning also the private key, from electronic devices. And that's been done, you know, the issue of Tempest-style attacks using RF emanations from things has been known for decades. So that's not news. However, these guys have figured out how to do it in a single shot, which makes it vastly more effective than any previously known attacks. In other words, attacks never get weaker, they only get stronger.

So two weeks ago in Baltimore a team from Georgia State University suddenly made this style of attack practical in the real world. And it's worth noting this is a good lesson in why it's worth pursuing even weak vulnerabilities like Meltdown and Spectre arguably are, since we've never seen them in the wild, and as far as we know they've never been weaponized. But you have to say "yet" because, similarly, an attack which used RF, which you could only get to by opening the case of devices and putting a probe on the backs of the chips, which is pretty much what was needed until now - oh, and doing over and over and over, re-use those keys you're trying to extract.

You could argue, okay, yeah, so chips are going to leak some RF. Okay. But you've got to open the device, you've got to stick an antenna on the back of the chip, and then you've got to make it do its thing for hours while you collect all the data and process it. You could say, eh, not that exploitable. Well, in one jump we go to a single operation no longer requiring that the device be opened. You can get enough signal from the outside. Now this thing suddenly goes practical. The point is, impractical things can suddenly become practical, and we may very well see this with all these microcode problems of this incredible install base of Intel chips that we have now.

Their paper was chillingly titled "One & Done: A Single-Decryption EM [electromagnetic] Based Attack on OpenSSL Constant-Time Blinded RSA." In other words, this was a good implementation of RSA using constant time in order to prevent a side-channel attack based on how long the operation took. Their abstract reads: "This paper presents the first side-channel attack approach that, without relying on the cache organization and/or timing, retrieves the secret exponent from a single decryption on arbitrary ciphertext in a

modern, that is, a current version of OpenSSL, fixed-window constant-time implementation of RSA.

"Specifically, the attack recovers the exponent's bits during modular exponentiation from analog signals that are unintentionally produced by the processor as it executes the constant-time code that constructs the value of each 'window' in the exponent, rather than the signals that correspond to squaring/multiplication operations and/or cache behavior during multiplicand table lookup operations." So they basically used different information that allows them to do it in a single shot. They said: "This approach is demonstrated using EM emanations on two mobile phones and an embedded system; and after only one decryption in a fixed-window RSA implementation, it recovers enough bits of the secret exponents to enable very efficient - within seconds - reconstruction of the full private RSA key."

They said: "Since the value of the ciphertext is irrelevant to our attack" - meaning all they want is the key - "the attack succeeds even when the ciphertext is unknown and/or when message randomization, that is, 'blinding' is used." They don't care. "Our evaluation uses signals obtained by demodulating the signal from a relatively narrow band [40 MHz band] around the processor's clock frequency of around a gigahertz, which is within the capabilities of compact sub-$1,000 software-defined radio receivers."

They say: "Finally, we propose a mitigation where the bits of the exponent are only obtained from an exponent in integer-sized groups, for example, tens of bits, rather than, as is currently being done, obtaining them one bit at a time." They said: "This mitigation is effective because it forces the attacker to attempt recovery of tens of bits from a brief snippet of signal, rather than having a separate signal snippet for each individual bit. This mitigation has been submitted to OpenSSL and was merged into its master source code prior to the publication of the paper." And finally they said: "In our experiments we place probes very close, but without physical contact with the unopened case of the phone, while for the embedded system board we position the probes 20 centimeters away from the board, so we consider the demonstrated attacks to be close proximity and non-intrusive."

So anyway, I liked this as a perfect example of why something that has not been a problem suddenly jumps to becoming a problem. The good news is OpenSSL will have an update in its code as a consequence to prevent this. And to the degree that devices ever get updated - and again, lots of devices are never going to. So now what we have is a practical attack without contact, meaning just proximity. And certainly, for example, any router that is using OpenSSL's library for its work, it's in a plastic box, probably from the outside is going to have its emanations detectable.

And in their detailed presentation, which I won't and haven't gone into, they talk about how laying the phone on the counter at Starbucks or in any environment where the phone is in contact with a surface, what could be behind that surface is something close enough to, if the phone does any pre-patched OpenSSL work - and again, OpenSSL is the library that underpins certainly all of the open source crypto that we see. And even non-OpenSSL, yeah, non-OpenSSL, which is using the same bit-at-a-time approach that OpenSSL was previously using would also be vulnerable. So I just - we have to, I'll say it again, we have to continue to allow security researchers to poke at our systems because all they are doing as a consequence is making them stronger.

Leo: Oh, boy.

Steve: So this is not - I guess it could be worrisome depending upon your situation. OpenSSH, Open Secure Shell, which is super widely used...

Leo: Oh, I use it every day to log into my server, yeah.

Steve: Yes, yes. Since day one, since its birth several decades ago, for the past two decades, every version of OpenSSH server has had a worrisome security flaw.

Leo: Oh, no.

Steve: Yeah. Now, again, not probably terminal. But the title of the problem posted to seclists.org is "OpenSSH Username Enumeration." In examining an unrelated source code fix and commit, researchers at Securitum.pl realized that a potentially bigger problem had also been inadvertently fixed. So again, the reason it's so valuable to have open source and eyeballs looking at it. They wrote: "We realize that, without this patch, a remote attacker can easily test whether a certain user exists or not, thus enabling username enumeration on a target OpenSSH server." In other words, username without password. And as we've talked about often, what you really don't want is a website that says, first of all, who are you, and then says we know you or we don't, before asking you for your password. The proper design is give me your username and your password, and then we're not going to tell you, if we're not happy, why we're not happy. We're just going to say one of those is wrong.

And in fact we saw sort of something just like this with the failure of the WPS protocol on WiFi. Remember that it was supposed to be eight digits. It was really seven because the last digit is a check digit which could be computed from the other seven. But even seven would be pretty secure. Except that it turns out you could chop it in half. You could separately check the first and then check the second. Well, this is a little bit like that. You can separately check for usernames. And then, once you've got the username, then you're able to pound on the password. So it does arguably reduce the security of an OpenSSH server.

Leo: I use a public key login.

Steve: And I would never do anything else. I do the same thing. Oh, absolutely.

Leo: Okay. So this isn't vulnerable if you're using public/private key logins.

Steve: Correct.

Leo: Okay.

Steve: So what they said is the attacker can try to authenticate a user with a malformed packet. And they literally - essentially they truncate the packet. What they discovered is the return from a malformed packet is different if the username is known versus if it's not. And so based on that differential error that is returned, that allows the server to be probed. They said: "We believe that this issue warrants a CVE. It affects all operating systems, all OpenSSH versions." They said: "We went back as far as OpenSSH 2.3.0, released in November 2000," but as far as they know it's always been there. And they said: "...and is easier to exploit than previous OpenSSH username enumerations, which

were all timing attacks." Previous ways of doing this, you know, you would get a slightly different timing, so you'd have to repeat it in order to disambiguate network-based timing. And this is much easier.

So they said: "We also believe that this should be posted to oss-security right away. The issue, that is, the commit, is already public. And if we spotted it, then others not so well intentioned did, too. We are at your disposal for questions, comments, and further discussions." So anyway, again, not super worrisome. And I'm like you, Leo, both with OpenSSH and also, for what it's worth, with OpenVPN. I'm only using certificates in order to do the authentication because that's much stronger.

And get a load of this. We old farts know about AT commands. You know, you type ATA and hit Enter, and then out on the terminal says okay. And then you proceed from there, you know, ATDT and so forth, for dial touch tone. Yeah. Turns out a final gift from the Baltimore USENIX conference was a paper titled "Attention Spanned: Comprehensive Vulnerability Analysis of AT Commands Within" - believe it or not - "the Android Ecosystem." AT commands are not gone. In fact, they have quietly been hugely expanded.

A multisource team of 11 researchers from Samsung Research America, Stony Brook University, and U of Florida took a close look into the undocumented AT commands currently supported on contemporary Android devices, and what they found was surprising and a little worrisome. The team analyzed and reverse-engineered the firmware images of more than 2,000 Android devices from 11 major Android OEMs: ASUS, Google, HTC, Huawei, Lenovo, LG, LineageOS, Motorola, Samsung, Sony, ZTE, and others. They discovered that these devices support over 3,500 different types of AT commands, many of which grant access to very dangerous functions.

Now, of course you've got to have a web domain; right? ATcommands.org, Leo. Nice-looking graphics and presentation. ATcommands.org. The abstract of their paper reads: "AT commands, originally designed in the early '80s for controlling modems" - which is what we were all using them for - "are still in use in most modern smartphones to support telephony functions." Remember how we've talked about how there's a baseband processor which is really obscure and, it is believed, as a consequence of lack of lots of oversight, probably full of problems. The processor that we interact with, with a smartphone, is not the baseband processor. It's a separate subsystem. And AT commands are often used for them to talk to each other.

So their abstract says: "The role of AT commands in these devices has vastly expanded through vendor-specific customizations." Because, like, okay, there's the channel, it's established, and developers are just adding to what's already there. Anyway, they said: "Yet the extent of their functionality is unclear and poorly documented." And then I put in here parenthetically: In other words, deliberately undocumented.

They said: "In this paper, we systematically retrieve and extract 3,500 AT commands from over 2,000 Android smartphone firmware images across 11 vendors. We methodically test our corpus of AT commands against eight Android devices from four different vendors through their USB interface and characterize the powerful functionality exposed, including the ability to rewrite device firmware" - get that, yes, an AT command can rewrite device firmware through the USB port - "bypass Android security mechanisms, exfiltrate sensitive device information, perform screen unlocks, and inject touch events solely through the use of AT commands." And who wants to bet that that's not the way the companies like Cellebrite are getting up to some of their games, is they already know this, yet they're not exposing it to daylight the way security researchers are.

They say: "We demonstrate that the AT command interface contains an alarming amount of unconstrained functionality and presents a broad attack surface on Android devices, and all available through most devices' USB port."

So I've got here in the show notes ATcommands.org is the site, for anyone who's interested. And ATcommands.org/atdb/vendors will take you to a page where you can check your own phone, if you've got an Android device, your own phone and what they found about your vendor and phone model.

Leo: I'm baffled that you would support AT commands.

Steve: I know.

Leo: It's bizarre.

Steve: I know. And it's still in use today because it's always been there; and they just said, oh, well, let's just add some more proprietary AT commands. And so, like, the screen, when you do touches on the screen. And maybe it's part of production testing, like you plug the device into a special USB...

Leo: Oh, yeah, because you can have a serial port kind of controlling the testing, yeah.

Steve: Right. And in order to simulate touch events. But they ship it that way, Leo.

Leo: That's nuts.

Steve: So then they have a little Q&A. "How does this affect me?" They say: "On some Android smartphones, an AT command interface is exposed over USB without USB debugging enabled." Meaning generic. "Unfortunately, some devices do not authenticate this interface or allow it to be used from the lock screen. We found that in some cases the 'charge-only' USB mode may also fail to block AT commands." So even when you have it set to only allow charging, still happily chatting away over the AT commands. "This means unsuspecting users who plug in their phones to a USB port for charging..."

Leo: Oh, my god.

Steve: Uh-huh.

Leo: So even if I have that condom you gave me, I could still get - no, because that's only power.

Steve: No, no, no, that's only charging, yes.

Leo: You'd still have to have data lines, okay.

Steve: Yes, yes. But if you plug it in just because you want a charge, it can be malicious and reach into your phone and do a full takeover. "This means," they said, "unsuspecting users who plug in their phones to a USB port for charging or data transfer may have their devices locally compromised by a possibly pre-recorded sequence of AT commands. Furthermore," they write, "many commands, such as those for exfiltrating sensitive data, have no visible side-effects." In other words, the phone looks just fine. You don't know what's going on underneath.

Leo: Oh, my goodness.

Steve: So they said - I know. They said: "Have you found any vulnerabilities? Yes. We have notified each vendor of any relevant findings and have worked with their security teams to remediate the issues." So not only are their devices wide open, but they're not even secure. They've got vulnerabilities in their handling of the AT commands.

And then they finally said: "Did you find any remotely exploitable vulnerabilities?" They said no, but listen to the caveat. "All of our investigation centered on the device's USB connection. We did not investigate remote AT attack surface. But the first places we would look would be the Bluetooth interface and the baseband." Meaning over the air.

Leo: Oh, man.

Steve: Yikes.

Leo: I mean, you're doing keystrokes. You're doing...

Steve: Yes.

Leo: Basically you can interact with the interface in any way possible.

Steve: Complete remote control of the phone.

Leo: Wow.

Steve: By plugging it in.

Leo: And even where it says "Do you allow this?" you can click the "Allow" button.

Steve: Yes.

Leo: So the fact that the UI says, "Are you sure?" No, it's like UAC, you can just click it.

Steve: That's right.

Leo: Holy cow. Oh, man, that's terrible.

Steve: Yes. So probably worth checking. And again, for phones that are current, this comes back to what you and I were talking about when we were talking about the man in the disk and the similar things. If you're going to go Android, go big. Go with somebody like a Samsung or a Google who are actively staying on top of the security of their devices because they're neat; they're very capable; they're more open than Apple phones on the iOS platform. But there is a serious tradeoff for vulnerability. The good news is these guys talk to the vendors, and the vendors will likely respond. But wow. Nice piece of research. And, whoo, worrisome.

I've a bit of miscellany. Ryan Dodds, he wrote on the 17th: "@SGgrc Steve, did you ever find the perfect coffee travel cup?" He said "cup." I'll go with "container." He says: "I'm struggling. All travel cups are usually plastic and make the coffee taste awful." And so Ryan, and any other listeners, there's only one word that I have, and that's Contigo. And that seems to be the universal answer to that question. Contigo is a company, C-O-N-T-I-G-O. That's what you want.

They do make some plastic travel containers for cold beverages, but they make absolutely, end of story, the best hot coffee transport ever. They've got a really good closing mechanism. You're able to squeeze it, and it opens. I don't even know how it works. It's so good. It just absolutely doesn't drip, but it opens an air hole and the coffee drinking hole at the same time. Contigo, C-O-N-T-I-G-O. They're also of course double-lined vacuum containers so they keep the coffee hot for hours. Anyway, that's what you want, Ryan, and anybody else.

Leo: What happened to the Temperfect Mug?

Steve: Ah, it was a gimmick. You know. In fact, did it ever come? I guess it did.

Leo: Yeah, we both got them. I use it once in a while. You can't dishwash it. So it's kind of...

Steve: Ah, that's a problem.

Leo: I mean, that's the problem for me, yeah. I use a Contigo, as well. I like the Contigos.

Steve: Yeah. And Temperfect had all this problem with that shutter. That's a real problem. That shutter thing, you know, it just doesn't work very well. So it's like, eh, no. Contigo is what everybody wants. And the other thing everybody wants, not surprisingly, is a copy of SpinRite.

Leo: As they should.

Steve: As they should. Floris, who tweeted a little earlier this month, I just saw his tweet, said: "Just had to help a company figure out moving their super low RPM old half-broken 100-plus hard drives on slow USB1 and USB2 to a new Thunderbolt RAID system." He says: "I have a feeling I will be using @SGgrc's SpinRite a lot next week." He says: "Time to push that company to buying a proper license." And Floris, in this instance, yeah. If you've got more than a hundred hard drives, that kind of pushes my extremely tolerant, give it a try, fix your mother's or your friend's hard drive, and I'm not going to care. A company that's got more than a hundred drives, and you're moving them, you're resurrecting them and checking them out and moving them over to Thunderbolt RAID, I think they can afford four copies. That's the deal.

An individual buys one copy. A company, we like them to have four in a so-called "site license." That seems fair since we're letting them use SpinRite forever on all the systems that they have, either spinning or solid-state, because we know that it repairs both. And I kind of came up with that because that way if someone can buy one, if it works, then they can buy just three more. They don't have to say, well, you know, I already got one, so how do I buy the corporate license, blah blah blah. As long as you maintain four current licenses, then you're good to go with a corporate site license.

And whenever I heard three yabba dabbas, I'm happy because that says, hey, somebody got one, they tried it, and then they said, hey, we want to be licensed to use it as a company. So I always appreciate hearing that. So thank you, everybody, for doing that. And Floris, thanks for helping this company. And do give them a little push because a couple hundred drives, or more than a hundred, that's probably worth a license.

Leo: Time to buy one, yeah.

Steve: And that's our podcast, my friend.

Leo: What? What? How could it be over? Wait a minute. Wait a minute. What about never a dull moment? I guess there's no big story.

Steve: There was just all of these things.

Leo: All of the little ones added up to one big one.

Steve: Well, and frankly, as I was thinking about this, I probably should have moved the AT commands down to the end and made that - because it's really a biggie. But what the heck.

Leo: That's amazing. That one's terrible. I was just looking. The Pixel 2 phones are not, according to that site, vulnerable.

Steve: Correct. I did notice that, too.

Leo: Yeah, yeah. Whew.

Steve: And some of them are only vulnerable if rooted. So rooting is again a reason not to.

Leo: Steve. There's never a dull moment with Steve Gibson. You find him at GRC.com. That's where SpinRite lives, but also all sorts of other great stuff. It's really kind of a must-visit site for your geek travels: GRC.com. He's @SGgrc on Twitter. You can tweet him. You can direct message him if you've got a tip or question, a comment. You can also go to GRC.com/feedback to leave the question there. But I encourage you to go and get SpinRite, whatever else you do, because that is the ultimate disk recovery and maintenance tool. He also has podcast copies there, audio, and it's the only source for the transcriptions. So that's a good thing to have if you like to read along while you listen.

We have audio and video at our site, TWiT.tv/sn. We do the show every Tuesday, 1:30 Pacific, 4:30 Eastern, 20:30 UTC. If you want to watch, you can watch us do it live: TWiT.tv/live. You can also chat in the chatroom, irc.twit.tv with all the other people watching live. But on-demand copies are also available for you at TWiT.tv/sn or Steve's site. Or better yet, maybe even subscribe, audio or video. That way you'll have it the minute it's available, and you can keep a collection, have an entire archival system of all the episodes going back to Honey Monkeys.

Steve: And I'll note, Leo, that we do this even when you are on the Mediterranean because I am never on the Mediterranean.

Leo: You never take a break. We actually have to force Steve to take breaks for holidays. Yeah, it'll be Jason Howell filling in for the next three weeks on Security Now!, and I will be back on the 25th. So I won't see you for about a month.

Steve: That's going to be weird, but we'll still be here.

Leo: Nothing's going to happen. It'll be quiet.

Steve: That's right. The industry's going to take a vacation when you do. So that'll be just - that will be good.

Leo: No reason to have any security flaws while I'm on the ocean. Thanks, Steve. We'll see you next time on Security Now!.

Steve: Thanks, buddy.