Traitors in Our Midst

Description: This week, Leo and I discuss the best "I'm not a robot" video ever; Cisco's WebEx problem being far more pervasive than first believed; more bad news (and maybe some good news) for Netgear; Gmail adds .js to the no-no list; a hotel finally decides to abandon electronic room keying; more arguments against the use of modern AV; another clever exploitable CSS browser hack; some (hopefully final) password complexity follow-ups; a bit of errata and miscellany; a SQRL status update; a "Luke - trust the SpinRite" story; and a very nice analysis of a little-suspected threat hiding among us.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-597.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-597-lq.mp3

SHOW TEASE: Time for Security Now!. Steve Gibson is here. The latest security news, some fun videos, and the "Traitors in Our Midst" - something we all use that actually could be a foothold, a toehold for bad guys. It's coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 597, recorded Tuesday, January 31st, 2017: Traitors in Our Midst.

It's time for Security Now!, the show where we talk about the latest security news now - now! - with this guy here, Mr. Security in Chief, Steven "Tiberius" Gibson. Hello. Live long and secure.

Steve Gibson: Oh, it's a thumb out.

Leo: Thumb out.

Steve: The officially approved Spock Vulcan whatever it is, live long and prosper.

Leo: I was leaving the gym this morning, and the manager gave me the live long and prosper sign. And I thought, well, that's quite appropriate, actually.

Steve: Very nice.

Leo: Yeah. So how are you, Steve?

Steve: I am great. And we have another terrific-looking podcast. I ran across a really interesting - it started as a master's thesis and evolved into a formal security analysis, which led to my titling this podcast "Traitors in Our Midst."

Leo: Oh, my.

Steve: Yes. Something we've sort of touched on over the years, but we haven't, and the industry hasn't, to nearly enough degree. So that's our main focus, which we'll get to toward the end. But fully half of the tweets that I received over the last seven days were referrals to this hysterical video, the "I'm not a robot" arm, sort of defeating the purpose, or so everyone thinks. So on one hand, it's really funny. On the other, it actually, I mean, in some of the coverage, I think they've missed the point. But we'll get to that.

It turns out that Cisco's WebEx vulnerability that we talked about last week in the Chrome extension is far more pervasive than we first believed. More bad news, boy, it just keeps piling up for Netgear. But there's maybe a little light at the end of the tunnel. Gmail is restricting .js attachments as of middle of February, I think it's February 13th, which I assumed they were before, but they're going to start. There was some misreporting of a hotel, an old-school, 111-year-old Austrian hotel that was attacked by malware and locked the guests into their rooms. Well, that was never true, but...

Leo: Yes, I got bit by that. That was a fake news story, yeah.

Steve: Yeah. But there is some interesting news there. We've got some additional arguments against the use of modern AV from some people who really do know. Another, oh, you're going to love this one, Leo, a very clever exploitable CSS browser hack. Remember last week or the week before we showed how form fields that were being auto-filled could be hidden so that the user didn't see that they were being filled and sent back. There's one where you copy a command line that a web page is showing you into - you copy and paste. So you copy it from the browser and drop it into your terminal window, and it does much more than it looks like it does. In fact, I would tell people not to do that, but instead to drop it into Notepad. I did, and it's like, whoa, look what it actually is, that is, inside this very innocuous-looking little command. So we have that.

Some final follow-up, hopefully it's final, to the whole password complexity discussion we've been having the last few weeks. A bit of errata. Some miscellany. A SQRL status update that a lot of people have been asking for. A "Luke, trust the SpinRite" story. And a very nice analysis of a little-suspected threat hiding among us. So I think a great podcast.

Leo: Oh, how exciting. Hiding among us. Hmm. You've got a lot of spooky stuff in this one. Leo Laporte, Steve Gibson.

Steve: So our Picture of the Week is just - it's fun. It's crazy. It's not meant to be taken seriously. But it shows your typical wiring closet 19-inch rack, just crammed with cross-

connect cables. So it's like three square feet of RJ jacks, all plugged together with a big sign in the front, like on the glass on the outside: "In case of cyberattack, break glass and pull cables." Meaning just run away.

Leo: That would work, actually, wouldn't it? You just disconnect all those cables?

Steve: It would, yeah, although they don't look like they're labeled. And so you'd have a serious nightmare...

Leo: Just pull them all.

Steve: ...figuring out where the cables went, yes.

Leo: Where they go back afterwards, yeah.

Steve: It's like, wait, you took that seriously? No, we were just kidding.

Leo: Yeah.

Steve: And it's like, oh, I got a virus in my computer, so I pulled all the cables out. Oh.

Leo: Oh.

Steve: Okay. So for our listeners who won't be able to see this, while you were telling us about FreshBooks, I wanted to see whether googling "I am not a robot arm" would bring up anything so that we could tell our listeners who aren't seeing the video how to go see this for themselves. And, well, boy, does it - 4,100,000 hits on that phrase.

And the entire page of Google results: "Robot arm beats 'I am not a robot' CAPTCHA test," says Geekologie. Daily Mail: "Googly-eyed robot beats 'I'm not a robot' security system." The Enquirer: "Robot declares 'I'm not a robot.' World implodes." UPI: "Googly-eyed robot beats 'I'm not a robot,'" and on and on and on. "A robot arm successfully beats an 'I am not a robot' CAPTCHA," says LaughingSquid.com. And DailyMail.co.uk and on and on and no. So as I said, this thing, the reason it's here is that, first of all, there's been a global misconception about what this means, but also because it is just hysterical. So with that said, Leo, we need to share this video.

[Video plays]

Leo: Here you go. So should I describe it? There's a robot arm with a stylus doing a trackpad on a MacBook with one of those CAPTCHAS that says, you know, we've talked about it before, it's a Google thing, "I am not a robot," and you just click the checkbox. And the robot clicked it. I don't know how it found it.

Steve: Well, because the operator is, like, servoing it.

Leo: Oh, it's not autonomous.

Steve: No.

Leo: And it says, "Deal with it."

Steve: Okay. So, and the audio back there, I mean, it sounds like it's grinding gears. It's got two D cells or something, and it's just, you know, it's a toy. But very cleverly, the person who put this together got a capacitive stylus and used it to servo the cursor of the laptop over into the checkbox, and then lifts it up and then pushes it down. And as some have observed, it even does a mic drop at the end. It opens its jaws, drops the pen, and then takes a bow with its googly-eyes.

Leo: Okay.

Steve: So the reason, I mean, so what everyone has said is, oh, look. It fooled Google.

Leo: No.

Steve: It fooled the "I'm not a robot."

Leo: No. The laptop knows, it actually knows the operator.

Steve: Precisely. The only reason that option was presented is there is probably a multiyear-long preexisting relationship between that laptop and Google and the Internet. And so there's a massive history behind it which, as we've explained, is how this thing works, is that it's deep reputation based. And so, yeah, I mean, it's funny and clever, but all the press thinks this actually means that a robot beat the "I'm not a robot."

And so I wanted to make sure people understand, no, an Internet bot would have never received that option. It's because there's a person basically there. And even though he used a cheesy noisy arm to do what he could have easily done with his own finger, well, that doesn't really defeat the purpose. So I got a kick out of it. And, boy, it was the most shared thing with me of the week. So I wanted to share it with our listeners.

We talked last week about Tavis Ormandy's discovery of a mistake. You have to call it that because it is such a critical mistake that Cisco responded, to their credit, over the weekend. I think Tavis notified them on Friday and started his 90-day countdown, and he got to 89, and it was fixed. I mean, just immediately fixed. However, Cisco keeps coming back to the disclosure page, their security advisory page. And it was updated as recently as today because it turns out it's a little more pervasive than originally believed.

They said, quote: "A vulnerability in Cisco WebEx browser extensions [plural] could allow

an unauthenticated, remote attacker to execute arbitrary code with the privileges of the affected browser on an affected system. This vulnerability" - and here's where it gets much broader - "affects the browser extensions for Cisco WebEx Meetings Server and Cisco WebEx Centers [which is Meeting Center, Event Center, Training Center, and Support Center] when they are running on Microsoft Windows. The vulnerability," they explained, "is due to a design defect in an application programming interface (API) response parser within the plugin." Which is a nice way of saying whoever wrote this, whoever coded this, wasn't thinking about the way it could be abused. They were just thinking about getting it to work.

"An attacker that can convince an affected user to visit an attacker-controlled web page or follow an attacker-supplied link with an affected browser could exploit the vulnerability. If successful, the attacker could execute arbitrary code with the privileges of the affected browser. Cisco has released software updates for Google Chrome, Firefox, and Internet Explorer." So all three major browsers. All we knew last week was from Tavis's position because he was looking at Chrome, his baby that he's more responsible for. We didn't hear anything then about the other browser extensions. It turns out there's common code that Cisco reused across their third-party extensions, and all of the browsers are similarly affected, that is, Chrome, Firefox, and IE, except for Edge on Windows 10, which is effective.

So I heard, I received in response to our discussion last week a number of people whose companies used to use WebEx but did no longer, yet of course no one had gone in to retroactively remove it because it was there and didn't seem to be hurting anything. Well, now we know that that was an unused extension presenting an expanded attack surface. So standard best security practice, as with, for example, Java, is if you don't know you need it, if you're not actively using it, thereby justifying the unavoidable exposure that anything extra brings, it's just better to remove it.

So Cisco has updated all of their extensions. If you know that you need WebEx, and you're using either, as we said last week, Chrome, but also now we know Firefox or IE, since this has been getting a lot of attention in the industry, and it turns out it's not difficult to exploit, you want to make sure that you're up to speed.

Cisco did provide a workaround, suggesting that if something prevented updating the extensions, or for example in a large organization, IT could provide some filters at the border to prevent WebEx activity on non-WebEx sites, which is where the bulk of the problem comes from. Remember that Tavis said he hoped that the Cisco site didn't have any cross-site scripting vulnerabilities that would allow someone to loop their attack through that site, which would then bypass this protection.

So updating the extensions, and if you're an IT individual with responsibility for a network that maybe you don't have any per-machine control over, or your company needs to use WebEx, and you're not sure everybody's going to update, you can restrict the access so that the WebEx API can only be used on the sites where you expect it to be used, and not as an unintended attack vulnerability. And they said under "Workarounds": "There are no workarounds that address this vulnerability. However, administrators and users of Windows 10 systems may utilize Microsoft Edge to join and participate in WebEx sessions as Microsoft Edge is not affected by this vulnerability." So that's good news for any corporations that have moved to 10.

Netgear. Boy. You know, I think it's probably easily the top of our list of the most recently troubled router firmware. This is a vulnerability which affects at least tens of thousands, and perhaps hundreds of thousands of routers. The guy who wrote this up, Simon Kenin, posted to the SpiderLabs Blog at Trustwave. It was well written, sort of in a

first-person narrative, and kind of fun, so I'm just going to share it as he sort of discovered this.

He said: "It was a cold and rainy winter night, almost a year ago, when my lovely Netgear VEGN2610 modem/router lost connection to the Internet." He writes: "I was tucked in bed, cozy and warm. There was no way I was going downstairs to reset the modem. 'I will just reboot it through the web panel,' I thought to myself. Unfortunately, I couldn't remember the password, and it was too late at night to check whether my roommates had it. So I considered my options: Get out of bed, go downstairs and freeze as I reboot the router" - which itself I guess had frozen. "Or be lazy, stay in bed and, since I am a security researcher, try to hack it. Needless to say, I chose the latter.

"So where do I start, I thought to myself. Well, it has a web interface, and I need to bypass the authentication somehow, so the web server is a good start. I started manually fuzzing the web server with different parameters. I tried the '../..' classic directory traversal and so forth. And after about a minute of fuzzing, I tried '...' and got this response." And then in his explanation he shows us essentially a web page response with <html> section, then <head>, then <meta>, then <title>, then <body>, and then the close tags. And essentially it's a response, a 401 Unauthorized response, meaning that there's a problem with authorization or authentication, you know, password. It says: "Access to this resource is denied. Your client has not supplied the correct authentication." And then there's a form with a post, so you post something. And the action is "unauth.cgi?" and then an ID with a decimal number, 1211868232.

And so he says: "Hmm, what is that unauth.cgi thingy? And what does that ID number mean? Luckily for me," he writes, "the Internet connection had come back on its own, but I was now a man on a mission. So I started to look around to see if there were any known vulnerabilities for my VEGN2610. It turned out there were none. I started looking up what that 'unauth.cgi' page could be, and I found two publicly disclosed exploits from 2014 for different models that manage to do unauthenticated password disclosure. Booyah," he writes, "exactly what I need." And then he provides two links in his discussion of those two reports. "Those two guys found out that the number we get from unauth.cgi can be used with passwordrecovered.cgi to retrieve the credentials. I tested the method described in both and, voila, I have my password. Now I can go to sleep happy and satisfied.

"I woke up the next morning excited by the discovery. I thought to myself, three routers with same issue?" Meaning his and those two others. "Coincidence? I thought not. Luckily, I had another, older NETGEAR router laying around. I tested it and, bam, exploited. I started asking people I knew if they had Netgear equipment so I could test further to see the scope of the issue. In order to make life easier for non-technical people, I wrote a Python script to test for this issue.

"Now, I'm not a great programmer. I'm aware of that, and that's why I don't work as a full-time programmer. As it turned out, I had an error in my code where it didn't correctly take the number provided from the unauth.cgi, and it passed gibberish to passwordrecovered.cgi instead. But somehow it still managed to get the credentials." Meaning that you didn't even have to give passwordrecovered.cgi whatever that thing was, that ID string from unauth.cgi.

So he says: "Wait. What's going on here? After a few trials and errors trying to reproduce the issue, I found that the very first call to passwordrecovered.cgi will give out the credentials" - meaning the username and password - "no matter what the parameter you send. This is a totally new bug that I haven't seen anywhere else. When I tested both bugs on different Netgear models, I found that my second bug works on a much wider

range of models. A full description of these findings, as well as the Python script used for testing, is available online" - now, again, remember, this was almost a year ago - "and the vulnerabilities have been assigned CVE designations."

Then he says, with the subtopic "The Responsible Disclosure Process," which was a bit of a mixed bag, unfortunately, with Netgear, he says: "This is where the story of discovery ends and the story of disclosure begins. Following our responsible disclosure policy, we sent both findings to Netgear in the beginning of April 2016." So, what, nine months ago. "In our initial contact, the first advisory had 18 models" - 18 Netgear models - "listed as vulnerable, although six of them did not have the vulnerability in the latest firmware. Perhaps," he wonders, "it was fixed as part of a different patch cycle. The second advisory included 25 models, all of which were vulnerable in their latest firmware version."

So that was April. So two months later: "In June, Netgear published a notice that provided a fix for a small subset of vulnerable routers and a workaround for the rest. They also made the commitment to working toward 100% coverage for all affected routers. The notice has been updated several times since then and currently contains 31 vulnerable models, 18 of which are patched now, and two models that they previously listed as vulnerable, but are now listed as not vulnerable. In fact, our tests show that one of the models listed as not vulnerable" - which was a DGN2200v4 - "is, in fact, vulnerable, and this can easily be reproduced with the proof-of-concept provided in our advisory.

"Over the past nine months we attempted to contact Netgear multiple times for clarification and to allow them time to patch more models. Over that time we have found more vulnerable models that were not listed in the initial notice, although they were added later. We also discovered that the Lenovo R3220 router is powered by Netgear firmware and is vulnerable, as well. Luckily, Netgear did eventually get back to us right before we were set to disclose these vulnerabilities publicly. We were a little skeptical since our experience to date matched that of other third-party vulnerability researchers who have tried to responsibly disclose to Netgear, only to be met with frustration." And of course that's some of the stories we've been discussing the last few months.

"Two changes helped sway our opinion. The first was that Netgear committed to pushing out firmware to the currently unpatched models on an aggressive timeline. The second change made us more confident that Netgear was not just serious about patching these vulnerabilities, but serious about changing how they handle third-party disclosure in general. That change was their commitment to Bugcrowd" - and then he provides a URL, bugcrowd.com/netgear - "a popular," he writes, "third-party vendor that helps to vet research, provides oversight for the patching process, and provides bug bounty rewards to help to motivate third-party researchers. We fully expect this move will not only smooth the relationship between third-party researchers and Netgear, but in the end will result in a more secure line of products and services."

Anyway, and then he concludes, asking himself rhetorically why is this vulnerability so critical. He says: "For starters, it affects a large number of models. We found more than 10,000 vulnerable devices that are remotely accessible. The real number of affected devices is probably in the hundreds of thousands, if not over a million. The vulnerability can be used by a remote attacker if remote administration is set to be Internet facing. By default, this is not turned on. However, anyone with physical access to a network with a vulnerable router can exploit it locally." And of course that would also mean they could turn it on to then make remote exploitation possible.

Ah, and "This would include public WiFi spaces like cafs and libraries using vulnerable

Netgear equipment." Because by definition all of those people are on the inside of the router. Normally you would depend upon the router's admin username and password to protect it from unknown users on the LAN side. This bypasses that, making this an important thing to fix. And he says, "As many people reuse their password, having the admin password" - this doesn't just bypass it, this gives it to you. So you know what the password is.

"Having the admin password of the router gives us an initial foothold into the network. We can see all the devices connected to the network and try to access them, perhaps with the same password. With malware such as the Mirai botnet being out there, it's also possible that some of the vulnerable routers could be infected and ultimately used as bots, as well. If running a bot is not possible, the DNS can be easily changed to a rogue server, as described" - and of course we've talked about DNS exploits before.

So he finishes: "We recommend" - and certainly I do, too - "that all users of Netgear equipment check the Knowledge Base Article for instructions to test if they're vulnerable and/or how to apply patched firmware if they are." So we've talked about Netgear enough that hopefully any users are staying on top of their firmware. Again, you'd have to deliberately turn on remote admin, but you may be depending upon the router's authentication, username and password, to protect you. That is bypassable trivially in the case of these vulnerabilities.

And I know I ran through this quickly and threw a lot out there, but as I read this, there are still to this day a large population of routers vulnerable. So at this point you'd have to say that remote admin is unsafe on this family of routers, and come up with some other way to get the job done without remote admin, unless you're able to confirm that you're not vulnerable. And of course the problem is there's been now such a history of vulnerability in this family of routers that I just don't think you can consider it safe to set up a Netgear with remote admin. Who knows what other problems are already there, may already be known. Yikes.

I was surprised that Gmail was still allowing .js attachments, that is, JavaScript attachments. But they announced that, and they gave some warning, that as of February 13th, Gmail will start blocking and bouncing any incoming mail with JavaScript attached. They currently restrict a wide array of extensions, like .bat, .chm, .com, .exe - the ones you would expect - .msc, .vbe, Windows scripting host, .sys, a whole bunch, .lib. But they had not had .js on the list before. Oh, and .scr, because as we know that's a Windows screensaver which is actually an EXE which is just renamed .scr to differentiate it.

And of course also we know that they will look inside of different forms of compression, and both .gz and .bz2, .zip and .tgz files. And if they see files with those extensions in the zips, they block that. And if they see a password-protected archive that they're not able to look into, they block that, too. So I'm glad that Google is raising the barrier. I just wanted to give people a heads-up that - I don't know why anyone would attach a JavaScript to email. I guess you might stick it in a .zip file and get it to somebody if you had some cause for doing that. But you won't be able to.

And they said, if you still need to send .js files for legitimate reasons, and obviously the reason they're blocking it is not because they're grumpy, but because you just get up to too much mischief if you attach JavaScript to email. They say you can use Google Drive and Google Cloud Storage or other storage solutions, and of course just forward a link to somebody who can then grab the copy that you posted somewhere.

Okay, this hotel story. The reporting was fun because it said people were getting locked

in their rooms, which never happened. And so it may have been some confusion about language. It's an Austrian hotel that's been around for 111 years. And the problem is it has been plagued with cryptomalware which has three times in the past infected the hotel's networks, including the door keying system, which has prevented people from getting into their rooms.

So there is some sort of - the way their system works, obviously, is that you don't need the computer just to create the key, but the computer is somehow involved, some room management computer is involved in the unlocking process such that, if that computer goes down, all the rooms are locked. People can still leave. You just can't get back in.

So the managing director of this hotel - it's a four-star hotel in Austria, very high-end - his name is Christoph Brandstaetter. And he was quoted in the coverage saying: "The house was totally booked with" - now, this is in the most recent attack - "with 180 guests. We had no other choice. Neither police nor insurance help you in this case. The restoration of our system after the first attack in summer has cost us several thousand Euros. We did not get any money from the insurance so far because none of those to blame could be found." He said it was cheaper and faster for the hotel to just pay the Bitcoin.

Brandstaetter said: "Every euro that is paid to blackmailers hurts us. We know that, and we know that other colleagues have been attacked who have done similarly." He said: "When the attackers got the money, they unlocked the key registry system and all other computers, making them run normally again." So the conclusion of this is sort of sad. He said: "We are planning at the next room refurbishment for old-fashioned door locks with real keys, just like 111 years ago at the time of our great-grandfathers."

Leo: Can't hack them.

Steve: So here's a situation where they thought they would move up with the times and stay current and do the right thing. But they just keep getting hacked. Four times. And so there have been three previous hacks, and I guess there was a fourth attempt that was thwarted. But they said, okay, look, that's enough of this. We cannot be going through this and have our guests inconvenienced.

So, yep, going back to old-school keys. It'll be kind of quaint, but it's also sad. And you have to think, too, that this is certainly going to hurt the vendor who sells the system because they're going to get a reputation among the hotelier community of having a system that is attackable and that people have to stop using because it creates an actual problem for guests. And it'd be a little frustrating if you couldn't get into your room. I mean, what do you do, if you've switched everything over to electronic keying and that system breaks? So maybe a sign of the times.

Leo: So the only thing that was made up was that people were stuck in the room. The hack happened. They couldn't create new card keys. The old ones didn't work.

Steve: Correct.

Leo: But nobody was stuck in the room, which would make sense.

Steve: Nobody was trapped, exactly.

Leo: Yeah, yeah. You'd have to have a pretty - it'd be more like a jail cell than a hotel room, if you couldn't...

Steve: Yeah, and I'm sure that it would even violate all kinds of codes.

Leo: Safety codes, yeah.

Steve: Yeah, there's no way that ever able to [crosstalk].

Leo: I should have thought about that, yeah. What if the power went out? People wouldn't be able to get out of their rooms? No, that wouldn't be good, yeah.

Steve: Right. So an interesting story appeared on Friday in Ars Technica, written by Sebastian Anthony in the U.K., when a former Firefox developer, Robert O'Callahan, who is now a free agent and safe from, as Sebastian put it, the PR tentacles of his corporate overlord, said that antivirus software is terrible, AV vendors are terrible, and you should uninstall your antivirus software immediately - unless you use Microsoft's Windows Defender, which is apparently okay. And this has been a growing trend. A couple of months back, Justin Schuh, Google Chrome's security chief, and indeed one of the world's top "infosec bods," as Sebastian put it, said that antivirus software is - okay. This is Google Chrome's security chief quoted saying: "My single biggest impediment to shipping a secure browser."

Leo: Wow.

Steve: Yeah. Further down the thread he explains that meddling AV software delayed Win32 Flash sandboxing for over a year, and that further sandboxing efforts are still on hold due to antivirus. The man-in-the-middle nature of antivirus also causes a stream of TLS errors, says Schuh, which in turn breaks some elements of HTTPS/HSTS. And of course we've talked about that, how a number of the AV, in order to get visibility into HTTPS streams, they're now installing their own certificates in their users' machines and essentially spoofing the certs of sites you go to so that, if you go to a secure site and look at the certificate in your browser, it will say - it won't be from the site itself. It'll have been created by that AV software that you installed on your machine. So it's doing on-the-fly man-in-the-middle. And the point is it's not just kind of worrisome, but it's actively causing software vendors major trouble.

"These are just two recent instances," writes Sebastian, "of browser makers being increasingly upset with antivirus software. Back in 2012, Nicholas Nethercote, another Mozillian working on Firefox's MemShrink project, said that 'McAfee is killing us.'" Yeah. "In that case, Nethercote was trying to reduce the memory footprint of Firefox and found that gnarly browser add-ons like McAfee were consuming a huge amount of memory, among other things. If you venture into the browser mailing lists, anti-antivirus sentiment has bubbled away just below the surface for a very long time.

"The problem," writes Sebastian, "from the perspective of the browser makers, is that antivirus software is incredibly invasive." And of course that's why Windows Defender is an exception, because it's built into the OS. It's not a third-party add-on that is forced to do things that are unauthorized in order to hook itself in to the depth that it needs to. And then, just to finish what Sebastian wrote: "Antivirus, in an attempt to catch viruses before they can infect your system, forcibly hooks itself into other pieces of software on your computer, such as your browser, word processor, or even the OS kernel. O'Callahan gives one particularly egregious example."

He says: "Back when we first made sure ASLR" - remember Address Space Layout Randomization - "was working for Firefox on Windows, many AV vendors broke it by injecting their own ASLR-disabled DLLs into our processes." So we've discussed here that not only all of that, but remember that modern AV is also introducing a larger and more fertile attack surface. We've discussed, I think it was Symantec whose AV itself had buffer overrun errors. So you could take a system that had been carefully vetted and tweaked and developed over time, where there weren't any problems, there wasn't anything for the virus to attack, and add antivirus to it and make it vulnerable because of the AV, which was itself vulnerable.

So anyway, I thought that was very good coverage and some nice insight sort of behind the scenes. That's not a view we normally have. And there is some problem. The reason these developers aren't being vocal about this is that they recognize how popular AV is. They're trying to coexist. But it's very difficult. The AV people have, I mean, they're like an endangered niche in the market, which I think, certainly in the case of Windows, doesn't make as much sense as it did originally because, just as Microsoft finally added an actual firewall, and then had it there but it was turned off, then finally turned it on, now, I mean, you don't see third parties doing firewalls the way they used to. That finally went away.

Well, the AV market is hanging on because people see it as insurance. It's like, well, you know, I don't want to get infected, so I'll add this. But it turns out that it can make your system more vulnerable, rather than less vulnerable. And we've seen instances.

Leo: Yeah, wow.

Steve: Okay, now, Leo. This link, this look-before-you-paste. You've got to go to that page, and then you'll see this nice little command, "ls -lat," which obviously is Linux's or Unix's directory list command. Copy that, you know, mark and copy, and then go to a Notepad and paste what you've copied. That cute little, what is it, one, two, three, four, five, six, seven characters, "ls -lat."

Leo: I'm just going to copy that right off this page right here.

Steve: Copy it right off the page.

Leo: Copy that there.

Steve: Copy that.

Leo: Yeah. Now it's opened a terminal here because I think that's - I'd like to paste that command because apparently I don't know how to do an ls. But that's okay. Let's just paste this command here. Whoops. Paste that command. Whoa. I just gave my access - what the - hey, what happened? A lot of stuff happened there. What was all that that just zipped by? One of the problems with paste is it also pastes a return, so you can paste in and execute a command at the same time.

Steve: Right.

Leo: Let me just open the StickyPad or something else to look at that one because...

Steve: Yes, take a look at what is in...

Leo: That was a bad idea.

Steve: Take a look at what is in your...

Leo: Nothing bad happened. You wouldn't do that to me; would you?

Steve: No, I would not do that to you.

Leo: But fortunately...

Steve: And this is just meant to be a proof-of-concept. But it's very effective.

Leo: Wow.

Steve: So essentially this uses CSS, the technology for controlling the formatting of web pages, to hide a large block of code, which you can't see. And I don't know if anybody has experienced this, where you copy something out of a web page, like a high-end advertising-laden website. And when you paste it, you get a little ad stuck at the end of your paste, which is like, "If you are interested in additional information." And it's like, wait a minute, I didn't copy that. But it's, no, unfortunately, we've got scripting running, and we've got all kinds of technology that's able to override the normal behavior that we're used to with copy and paste. And so that's what's happening here is that cascading style sheets is used to take a big blob and just hide it from you.

Leo: Yikes.

Steve: Yes. And we've talked, for example, about how dangerous curl can be, where you just curl some URL, and you suck it in. Well, that could be used with this so that you see

the URL that you're going to curl. You first go look at it and go, oh, yeah, okay, that looks fine. Then you copy the curl command and the URL off of the page, which is actually different from what you see, and then run something coming from a different server and still get yourself blasted.

Leo: Wow.

Steve: So, boy, I tell you. What we're seeing is we've developed a very complex, very sophisticated set of interacting pieces. And unfortunately there's more and more attention being given to answering the question, what kind of mischief can we get up to by clever use, re-use of benign technology in mischievous ways. And of course, sometimes more than mischievous.

So I had a couple of follow-ups from last week. Someone sent via Twitter: "How strong do pronounceable passwords calculate?" And he says, "LastPass feature?" And he says: "I use for a few I must remember." And that sort of goes along with Max, who tweeted, he said: "You go on about logarithms for half an hour, but you fail to actually answer the question. God, that is so frustrating to listen to."

Okay. So, correct. The topic of last week was Password Complexity Calculation. That's what I delivered. I didn't give you password complexity conclusion, I gave you calculation. Because I wanted - my goal was to equip our listeners with a simple means for wrestling the questions of alphabet size, how many words in Diceware, how large is the alphabet, do I have upper and lowercase? How many characters, and so forth. How do you turn that into a meaningful number?

And so that's what we did last week. I stopped short on purpose because there isn't - I mean, okay. While there is a definitive means of saying passwords formed in this fashion will have this many equivalent bits of entropy, what people want is, well, how much should I have? How much is enough? And that's what there's no answer to. More is better. Less is worse. But the real problem is that we are ultimately in the hands of what the website or the system that we're providing that password to does with it.

If, for example, they do an MD5, you know, a lame old 128-bit hash for which all kinds of tables already exist, in fact it's like they just unsalted MD5. You have to have a really long, I mean, really, really high-entropy password in order for it not to be brute-forced because that hash doesn't provide enough brute-forcing strength. And tables can be applied because every user would be using the same hash. And so they would be able to essentially do a parallel attack. If, however, a more modern hash like an SHA-256 were used, that's good, except that unfortunately the crypto currencies use SHA-256, so now that's all been reduced to silicon, and it's screamingly fast.

But hopefully a forward-looking site, one that actually has good security, would be using a per-user salt, meaning that when you create your account, they just, out of the air, they pluck a random salt, and they put it into your account data. That is added to your password as salt to the hash in order to make every user's hash unique. So even two users with the same password would end up with different hashed results so that you wouldn't see that collision. And that prevents any kind of precomputation attack. And then, if we wanted to go really to where we should be, we use a password-based key derivation function, a PBKDF, where it's an iterative process that is deliberately intended to take 10,000, we've talked about this before, like 10,000 iterations. So you put this in, and you crank on it for seconds until you have an answer. So that dramatically reduces the ability to brute force.

But the problem is users have no control over that. We don't know what random website is going to do when we give it our password. So that's why I've been pushing back against this idea that there are easy solutions. "Easy" means low entropy. That's what's easy about it. If it's a word, that's low entropy. If it's an abbreviation for your favorite phrase, well, low entropy. We have no control. We don't know what they're going to do. So our best solution is to use the highest entropy raw input, with a reasonable length. Sometimes the website says no, no, no, that's too long.

Okay, well, so give it the highest entropy, longest password it lets us, and then store that somewhere. And, for example, this first person who tweeted, saying he uses a few of these pronounceable passwords that he must remember. Well, again, pronounceable means lower entropy. What LastPass had to do in order to make it pronounceable, and we sort of discussed this a few weeks back, where we were talking about that - what was that, oh, it was LessPass. And, boy, was it Less.

Leo: Oh, yeah.

Steve: Where, remember, each location in the password had a preassigned class of alphabet so that it was really lower entropy. So again, if it's pronounceable, less random. The point is convenience means lower entropy. The fact that you can remember it means it has lower entropy. So to this first person who said he uses a few, I'm not saying they're bad. I'm saying they're not as good. I'm not saying Diceware is bad. It's not as good. And then people say, oh, yeah, well, I use 27 Diceware words. It's like, okay. If you can find a website that will let you put 27 Diceware words in...

Leo: That'd be pretty good.

Steve: ...then go for it. I would never argue that that had low entropy, as long as you had a lot of words as candidates, and they were chosen at random. The problem is you can't find a website that will let you do that. So given that we generally have a length limit on our passwords, within that length I would always opt for as much entropy as possible. You're erring on the side of caution. And to the person who needs them to be pronounceable, so he can remember them, I say write them down. As Bruce Schneier said, we understand how to manage bits of paper. We have wallets. And leave off a couple of the leading or trailing characters so that it doesn't disclose the entire thing, in case somebody discovers it.

But you can write down a big long blob and then maybe customize it per site or something. There are some tricks you can do where you start with a root of high entropy, and anything you add to high entropy never reduces its entropy. So that's something important to remember. Anything you add to something with high entropy never reduces the entropy that you originally had. It can only add to it. It can only make it better. So anyway, it's been really useful to get some pushback and some feedback from people who, unfortunately, whose favorite practices I have disparaged a little bit, not intending to say you shouldn't do it, but just saying, okay, I want you to make an informed decision.

Leo: Right. If you're going to choose it, know how good it is.

Steve: Yes, yeah.

Leo: Now, let me ask you, because of course this is always the problem is the balance between memorable passwords and effective passwords. Memorable in general means it's less effective because it's less entropy. I try to balance it, and I wonder if this is a bad strategy, by using an algorithm to generate the password that I can recreate. For instance, I've mentioned in the past you could - this isn't what I use, and it's probably not long enough, but you could use the first initial of the last name of the last 10 presidents, capitalizing the Republicans. And if you wanted to add a digit to that, you could add the number of years they served. You could, in your head, go through that and regenerate it. But is that password inherently less reliable because there's an algorithm generating it?

Steve: Well...

Leo: I mean, it's memorable, so I guess it means less entropy.

Steve: Yeah, exactly. Although probably still high. Remember that the threat is brute-forcing.

Leo: So if somebody could figure out my algorithm or look at, you know, the worst thing would be they could look at it and go, oh, I see what he's doing.

Steve: Right. Exactly. And if someone didn't hash your password, and that escaped, and someone really was targeting Leo Laporte...

Leo: Well, I don't - reusing them is bad no matter what.

Steve: Well, no. But depending upon the algorithm, looking at it might give someone a clue to what it was you were doing. And, for example, they might notice, well, we don't know how he got these letters and numbers, but look at the pattern that's always the same.

Leo: Yeah, yeah. So it's a letter followed by a number, right.

Steve: Right, right.

Leo: So actually that's why I don't use it. But here's a good one. You could take page 10 of "War and Peace," and third paragraph in, and then just use the first letters of that paragraph plus punctuation.

Steve: And the problem is what that then does is encourage you to reuse that one phrase.

Leo: Well, no. Let's say I'm using it for my LastPass password. We all have one - if you're doing it right, you have one password that you have to remember, your Password Vault password.

Steve: Right.

Leo: Best just to make up a random one and memorize it; right? Force yourself to memorize it.

Steve: I really think that today there is no alternative than using a database, a password manager, to collect your passwords and just roll that way. I have all kinds of stuff stored in mine. And I just don't think there's a useful alternative, unfortunately. I mean, I know people want shortcuts and tricks. But the nature of that means that you're not going to get the security that you could have.

Leo: Yeah.

Steve: Two little bits of errata. We have listeners of the podcast among Google's security team. After last week they reached out...

Leo: That always makes me nervous when I hear stuff like that.

Steve: They reached out to add a bit of additional information about that troubling auto-form-fill hack, the one that we talked about where the fields were off the screen, and it was being populated. One of them wrote and said the CC number field, the credit card number field, will not be populated until and unless the user responds with the card's correct CVV. That's the card verification value. And so once again, tip of the hat to Google. They were thinking ahead.

So even with all the fields on the screen, Chrome will not populate the credit card number, even if it knows it. It waits for you to put in your three- or four-digit CVV to prove that you're you. And then it'll go, okay, and then, bing, populate the credit card field. So Google security team, thank you for the clarification. I'm glad to know that there's that extra safeguard in there. Of course, as we know, there's also street address and all kinds of other information that doesn't have the same protection.

And a good friend of the podcast and listener, Taylor Hornby at Defuse.ca, he shot me a note saying: "Quantum entanglement does not let you talk FTL" - as in faster than light - "but you can use it to get one-time-pad keys." So I just wanted to share that. There were a number of people who were sticking by Einstein's information cannot travel faster than the speed of light. That's an absolute limiting characteristic of the universe. Which China seems to think that's no longer a limitation. So we'll see because their system is definitely designed to allow people to experiment with FTL communications. But Taylor's note that you could use entangled photons to essentially securely feed two parties one-time-pad keys, that's a really good point, too. So Taylor, thanks for that.

Two bits of miscellany. Addam Tait wrote, he says: "Curious about your opinion regarding

software delivery teams. Better to have security person per team overseeing, or all devs taking ownership of security?" Which I thought was interesting. It's like, okay, so we're talking about security all the time, about what a problem it is, about how it doesn't seem to get the attention that it needs. So how do you structure a group of developers so that the product of their work is secure? And I think the answer is all of the above. That is, you make security an issue for all of them, that is, every single developer spends some time in meetings or in classes or somehow you just make sure this is on their radar. That is, that they not believe this is somebody else's problem.

And then also I don't think you can - you cannot avoid one person whose entire responsibility is security. That is the "buck stops there" person. And it's not that he's in the hot seat, he or she is in the hot seat. It's that it's so easy to get distracted when you're trying to get something to work that it's, okay, I'll check this to make sure I didn't make any security mistakes afterwards. Well, and then you get distracted by some other distraction, and you forget.

So I think, first of all, you need everyone on the team to be conscious of the importance of security. But you need it to absolutely be one person's sole focus so that everything that is produced is challenged from that perspective. That's that person's job. They're not distracted by anything else because that is, you know, that's all they have to do is make sure that the code being produced, the systems being produced are as secure as they know how to make them. I don't think - it just can't be part-time. Somebody has to have full-time oversight while everybody else understands the need.

And then, finally, Christian Loris said something that I thought was interesting, Leo, following up on my comment last week and our observation that "monkey" seemed bizarrely popular as a password. And he proposed something that I thought was kind of interesting. He said: "Heard your ponder on password monkey's prevalence. Maybe French because 'mon key' is French for 'my key.'"

Leo: Except it's not. But other than that, okay.

Steve: Oh.

Leo: It's a good guess. Key is "cl." "Mon cl" would by "my key."

Steve: Mon cl.

Leo: Mon cl.

Steve: Nice. Sorry, Christian.

Leo: Nice guess, though. No, I think people like monkeys. I don't know. It is a good question.

Steve: I think you're right. I think they're kind of cute and funny. And if you're not really caring back in the old days about security, it's, oh, monkey.

Leo: Monkey123 if you're really secure.

Steve: So I've had a lot of people saying, hey, Steve, you haven't been talking about SQRL lately. I confess that, as you know, Leo, I had gotten myself distracted for the last month.

Leo: The top-secret project.

Steve: But a couple weeks ago I announced to the SQRL newsgroup that I was back, that I'd done all I could for the time being on the distraction that I had been nursing. And I am back and active, and we're going to get this thing finished so I can get back to SpinRite.

But where I am at this instant I think everyone will find interesting. We're trying to make a practical system. And one of the things that SQRL does is it more tightly binds an identity to you. That is, as everyone knows, you set up one SQRL identity. That's sort of your surrogate on the Internet. And as you go to different websites, it presents your identity for that site, which is unique for everyone who uses that site. But the problem is, what about a site where Mom and Dad both want to do their banking?

Well, traditional username and password, we would call that "weakly bound," or "loosely bound," meaning that the only way the site knows them, knows anyone, is the username and password. So whichever side of the marriage sets up the banking account simply says to husband or wife, here's our username and password for banking. In other words, it's easy to share the identity that you're using because it is weakly bound. It is just a secret that you're trying to control.

Well, SQRL changes that. SQRL's identities are tightly bound. And not only does SQRL change it, but that's where we're headed. We're headed toward a world with more tightly bound identity. Biometrics does that. It's not easy to give your thumbprint to someone else the way you can give them your username and password. And if we get retina scans eventually, that'll be a problem. And arguably, the "relying parties," as they're called in the identity system, the relying parties want tighter binding. They want to know who it actually is and not just somebody who got told somebody else's username and password.

So although this isn't technically SQRL's problem, and it doesn't affect the SQRL protocol because the SQRL protocol is about asserting an identity, and this is about using those identity assertions, it still is a problem that needs to get solved. And anyway, so what we've developed is a solution we call "managed shared access," which I'm in the process of coding and will add it to the SQRL demo at GRC, mostly as a proof of concept, to demonstrate how in an environment where you have tightly bound identities, it's no longer practical to say, well, just share your username and password.

Now, without having this, we have a solution. That is, we had to come up with a kludge which is kind of ugly. And it's because I've never liked it that I thought, okay, there's got to be a better way. And of course the kludge is Mom and Dad each have their own SQRL identity, and then they also have a shared SQRL identity. So the banking site that doesn't support managed shared access, well, if it only allows one identity to log on, then it needs to be a shared identity. It can't be either one of their SQRL identities.

So they create a third, which they share. The reason this is messy is that quickly it gets

out of hand because then your corporation wants you to use SQRL to access the Intranet. IT controls it and, like, what, gives everyone a SQRL identity that that site recognizes. That's clearly not the right way to go, either. So the point is that there's a general problem with tightly bound identity not working well with the current world of loosely bound identity, which is really to say no particular assertion of identity at all. It's just, oh, yeah, here's the username and password.

So anyway, so we're in the process of - actually, I think we've pretty much figured out how we want this to work. And so I am adding that to GRC, not to say that anyone using SQRL has to do this, but I think by demonstrating an elegant, simple, low-overhead solution, which essentially allows an account owner to share access in a managed fashion with as many people as they need to for the application in a corporate environment, it might be everybody who's using SQRL with their own identities who needs access to a single relationship. So the point is, rather than creating a one-to-one binding, this allows a many-to-one mapping with management.

So that's where we are. So I'm working on that. And then it's just finish the installer, and I've got a little bit of UI stuff to wrap up. I mean, the system is running. Oh, there are a couple little issues involving the rekeying of identities that we haven't reached consensus on yet. But I expect we will soon. So I'm back in the saddle, and I'm going to push this thing across the finish line so I can get back to SpinRite.

And speaking of which, we have from Pete Kokkinis - K-O-K-K-I-N-I-S. He sent a note to Greg with the subject, "Please forward to Steve - testimonial." And this I referred to a little tongue-in-cheek as "Luke, trust the SpinRite." Because he said: "Hey, Steve. I'm a long-time SN listener and user of SpinRite." He writes: "You're an inspiration to me, and I look forward to your weekly podcasts. But for approximately 14 minutes I doubted SpinRite, and for that I'd like to apologize.

"I had a PC running at a client site acting as a syslogger for many years. Unbeknownst to me, my client closed up shop and moved when I went to visit their shared space with another tenant client of mine. So I picked up some things I had left behind, including the syslog PC. I fired it up back at my office only to find it hanging at boot-up with a message about hal.dll missing or corrupt." And, yes, many Windows users have, at least I've heard...

Leo: That's pretty much a showstopper right there.

Steve: Exactly. Yes, that's the hardware abstraction layer dot dll. And if you don't have that, you don't have Windows. So he says: "Being curious at this point, I ran SpinRite. Its forward progress stalled at 4%. Since I was impatient, I canceled it and tried booting as any running of SpinRite often seems to get enough of the disk back. This time it didn't. So I reran SpinRite, starting at 5%." Basically, so he skipped over the area that SpinRite was starting to really work on and work on recovering. He started at 5%, and it completed an hour later. "I tried booting again, nothing." Because of course he skipped over the work.

Anyway, "So I plugged the drive into a USB/SATA adapter to see if my syslog folder could simply be copied off. No dice. Even though I could see the folder, I couldn't even open it, as it would just hang. So I right-clicked the drive and ran a CHKDSK with both options checked." That's "Automatically fix file system errors" and "Scan for and attempt recovery of bad sectors."

He writes: "The drive was making a normal scanning noise" - and I kind of thought, okay, what is that? But anyway, scanning...

Leo: [Vocalizing]

Steve: Yeah, "as CHKDSK was running, and the progress bar was steadily moving. At this point I questioned the usefulness of SpinRite if CHKDSK can do this recovery more successfully. The progress bar finally reached the end after 14 short minutes, and I clicked Close. I browsed to the external drive letter, opened Program Files, and, bam," he says. "Half the folders in the system were completely gone, including the syslog folder I needed, which I had seen earlier.

"I googled if CHKDSK can delete files; and, wow, did I see some upset users out there. It never occurred to me that if CHKDSK can't repair, it will wipe data if corrupt, granted that you check both option boxes. Anyway, keep up the great work, and I will never doubt you again." So, and we do hear this often. People come to us having done this and saying, oh, oh, I didn't realize. Will SpinRite help me? And it's like, no.

Leo: Too late now, yeah.

Steve: No, Because what happens is CHKDSK doesn't do recovery. It just sort of says, okay.

Leo: It's all cleaned up.

Steve: This has gone into oblivion.

Leo: You don't want that.

Steve: Snip this off right here.

Leo: You didn't want that file, did you?

Steve: So it's like, it's just gone. And it's like, oh, sorry about that.

Leo: Sorry.

Steve: Okay. So are our ubiquitous printers traitors in our midst?

Leo: Oh, they are.

Steve: Yes, they are.

Leo: They are.

Steve: Yup. This began as a master's thesis titled "Exploiting Network Printers - A Survey of Security Flaws in Laser Printers and Multi-Function Devices." And this computer science grad who was working on his master's in computer science and network security took a very close look at printers. In the introduction to the paper he created, he wrote: "The paperless office has been a dream for more than three decades. However, nowadays printers are still one of the most essential devices for daily work and common Internet users. Instead of getting rid of them, printers evolved from simple printing devices to complex network computer systems installed directly in company networks and carrying lots of confidential data in their print queues. This makes them an attractive attack target."

And in this paper this person and two others basically take a very careful forensic look at 20 mainstream printers: HP LaserJet 1200, 4200N, 4250N, and on. There's, like, seven HPs, two Brothers, three Lexmarks, three Dells, a Kyocera, two Samsungs, a Konica, and an OKI. Every single one has a varying number, in some cases one, two, three, four, five, six, seven different classes of vulnerability. And so what we have, I mean, and this shouldn't surprise any listeners to the podcast, I mean, the details are interesting. Oh, in fact, the last column in the chart in the show notes is number of printer vulnerabilities they found. And in some cases, the two very popular HP LaserJets, the 4200N and the 4250N, both have 12 separate vulnerabilities each. Other ones have seven or 10, and seven, nine, 10, 10, five, 11. I mean, so rife with problems.

The problem is that security has not been traditionally a focus of our printers. And, I mean, and Leo, you obviously know this because you immediately knew the answer to the question, do we have traitors in our midst. I remember when Apple first produced the LaserWriter. You know, we were all tolerating Epson printers going bzzz, bzzz, bzzz, bzzz, bzzz, you know, back and forth. And suddenly, with no sound at all, out emerges this, like, photo-ready, shockingly attractive typeset piece of paper. It was astonishing.

And what I remember being really interested in at the time was recognizing that there was far more processing power in that LaserWriter itself in order to render the PostScript into bitmap than was in the machines that the printer was connected to. The machines could barely get DOS booted, yet they were able to, with the proper drivers, print a lovely page, thanks to the work that Apple did with HP. Or was it Canon back then? I don't remember. I think it was Canon initially.

So what we have is we have decades-old technology that has been quietly simmering and evolving with features being added and no one ever taking a close look at the security implications. And, for example, someone said, hey, let's do wireless printing. Oh, great. So then they add a wireless access point to this thing that is already layered with multiple languages and interpreters to turn the code into the printed page. And they give this thing an access point so that people with AirPrint can connect to it. What could possibly go wrong?

So they say in their paper: "In this paper we conduct a large-scale analysis of printer attacks and systematize our knowledge." And I ought to say this is the first time this has been done. The work is heavily referenced with all of the previous work, which has been scattered and, distressingly, is in some cases two decades old. There was some awareness of problems back in 1996 that still exist today, never got fixed. So they said:

"Based on our methodology, we implemented an open-source tool called PRinter Exploitation Toolkit (PRET). We used PRET to evaluate 20 printer models from different vendors and found all of them to be vulnerable to at least one of the tested attacks." And as I said, in some cases to 12 of the tested attacks.

"These attacks included, for example, simple denial-of-service (DoS) attacks or skilled attacks extracting print jobs and system files," meaning exfiltrating what the printer is printing. And think about that for a minute. I mean, think about what confidential information a printer has passed through it when anything that it prints could be exfiltrated from the network if something evil has managed to crawl into it.

So they say: "On top of our systematic analysis we reveal novel insights that enable attacks from the Internet" - from the Internet - "by using advanced" - get this, there is something called "cross-site printing techniques combined with printer CORS spoofing. Finally, we show how to apply our attacks to systems beyond typical printers like Google Cloud Print or document processing websites." For example, those that convert a PostScript into a PDF, for example. "We hope that novel aspects from our work will become the foundation for future researches, for example, for the analysis of IoT security."

So, okay. So the number one problem is what we would expect. People who've been following the podcast closely know that, in our current era, anytime you are interpreting something, you have a security challenge. When you have user-provided input to an interpreter, it is so difficult for an interpreter which essentially is reading what you provide as a form of code, it is so difficult to make sure there isn't some way to subvert that. And we've talked many times, for example, all of the Stagefright problems with the media library, those were problems in processing the media files because interpreters are used to read the tokenized formats of these files. They're convenient and efficient, but they're dangerous because they require the interpreter to be perfectly coded so that it cannot be subverted.

And so what we have with printers is, I mean, they are loaded with interpreters. That's what they do. By their nature they do not take a raw raster from the computer. They take a higher level description of the page. And that's what PostScript is. That's what PDF is. That's what PCL is, is a high level description. They read that, and then in their own internal RAM they build a bitmap which is then sent out to the printer imaging hardware as the page runs through it.

So, I mean, it is interpreter city in any of our contemporary printers. And they've got, in order to be maximally compatible, they all support - they're just bristling with protocols. There's the Internet Printing Protocol, IPP; the LPD, which is the Line Printer Daemon. There's SMB. A lot of them support Server Message Blocks. SNMP, which we've talked about, that's the generic Simple Network Management Protocol. Turns out all the printers support that, too. So you can query the printer. You can find out its make and model and all that remotely. And that of course tells you how to attack it. And then they also support, on port 9100, that's known as the JetDirect or AppSocket port, raw printing through port 9100.

So there's a wide array of protocols. These guys did not attack the protocols. Which is not to say they're not attackable, but they went for where the meat clearly was, which was the interpreters. There's a top layer which are the job or the printer control languages. Those are sort of the supervisors of the lower level page description languages. And so there's something called PJL, which is the Printer Job Language. PJL was originally introduced by HP, but soon became a de facto standard just due to HP's early strength in the printer market, in the laser printer market. It resides above the

other printer languages and can be used to do things like change settings, paper size, and the tray from which the paper will be pulled and so forth, sort of a meta level above page description. So PML, which is the Printer Management Language, which is a proprietary language also used to control HP printers, combines sort of PJL, the Job Language, with SNMP as its transport layer, in order to get that done.

Then page description languages are, you know, everybody can do their own if they want to because it only needs to be in agreement between the printer driver in the client machines and the printer itself. So consequently there's been an explosion of those. Kyocera has one called PRESCRIBE. Samsung has one that they call the Samsung Printer Language, SPL. Xerox has something called the Xerox Escape Sequence, XES. Canon has their own Printing System Language, CaPSL. Ricoh has the Refined Printing Command Stream, RPCS. Epson has the Standard Code for Printers, ESC. HP of course has HP-GL and HP-GL/2, which are primarily focused back in the days of plotters. So there's all of those. Then there's of course the PDF format, and most recently the XML Paper Specification, XML Paper Spec, or as we know it, XPS, which is often now found on newer printers. And we find support for that natively in Windows.

But among all of those, the most common standard page description language is PCL and then PostScript. PostScript is incredibly vulnerable, and that was one of the - actually most of the printers had problems with PostScript because it is a very complex language. It is a full stack, PostScript itself, a full, stack-based, Turing-complete programming language which consists of about 400 operators for math, stack manipulation, and graphic manipulation, and managing various types of data in arrays and dictionaries.

So, I mean, it's a language. And access to the PostScript interpreter, which is what all printers ultimately provide because that's how you provide, if you're using PostScript, that's how you provide them with a page description for them to interpret, access to a PostScript interpreter is classified as code execution because any algorithmic function can be implemented in PostScript. And PostScript has access to the file system because there have been extensions over time. Of course there have been. It's like, oh, wouldn't it be convenient if we could store things locally in the printer, and if we could get them back later?

So of course, yes, it's convenient, but it also means then that an attacker who has access to PostScript can use PostScript and the interpret either the way it was intended or by subverting it in order to gain access to the internals of the printer and in some cases set up long-term resident code, meaning that a printer in a network could be an advanced persistent threat, an APT, which as we know is one of the greatest concerns that corporations have. That's what brought Sony down, famously, was that something got into their system and lived there for a long time and allowed attackers to look around and gain a lot of knowledge.

So there were three attack models that these guys looked at. They looked at the local attacker, meaning physically local, that is, in physical proximity to the machine, who they characterize as the strongest attack possible, having physical access to the printer for even a limited time because that means that an attacker could plug in external storage media like memory cards or USB sticks; could temporarily connect the printer to a USB or parallel cable; and could change control panel settings. Essentially, that gives them a lot of power. On the other hand, it is realistic because printers are generally not seen as a security node in a network. They're typically, it's typically a shared resource in a public location.

They write: "It's a realistic attack for most institutions and companies. Gaining physical access to a printer can generally be considered easier than for other network components

like servers or workstations because they're usually explicitly shared by and accessible to a whole department. So sneaking into an unlocked copy room and launching a malicious print job from a USB stick takes only a matter of seconds."

Their second type of attacker doesn't have physical proximity, but is on the LAN. So a LAN-based attack can connect to the printer device via TCP/IP network, that is, the local LAN network, over any of these protocols - FTP, SMB, SNMP, LPD, and IPP, and even the raw port 9100. So there's lots of connectivity possibility in order to establish a connection. And they could do whatever they want to do over a longer period because their connection to the printer is part of what their own computer typically does during the day. So having that connection wouldn't even pop up in any kind of typical traffic analysis.

And as we mentioned before, many newer printers bring their own wireless access points to allow for easier printing and flexibility. AirPrint-compatible mobile apps are able to do that. And so while connecting to a printer through WiFi requires the attacker to stay physically close to the device, we know that it's possible to use focused antennas and that WiFi reaches a good distance these days, too. So you don't even have to be physically on the network in order to get a connection to an attacker.

And then, finally one of the more chilling attacks is - I mentioned this before - essentially a cross-site attack where a web-based attacker not in the network, not even connected to the printer, is able to gain access. And they demonstrate this in their work. It's actually possible to perform cross-site printing attacks.

So this so-called cross-site printing technique would be where a victim in a corporation unwittingly goes to a website and pulls a page from a malicious server that looks benign. Their web browser brings down a bunch of JavaScript and then uses Ajax requests to port 9100 of the victim's Intranet printer in order to talk to it, which is completely feasible. It is not, in this mode of attack, it is not possible to get results back because the same-origin policy in the browser will prevent an interaction with port 9100, but it is possible to send data to the printer. And so depending upon what's known about the attack, it's even possible for a remote web server to leverage JavaScript running in an Intranet user's browser to access the printer.

And I'm really glad these guys did this, pulled this all together, and that it's getting some attention. They did responsibly disclose everything they found to all of these manufacturers. I mean, in the case of HP, a score, scores of vulnerabilities were disclosed. It's not clear whether they have been fixed or will be fixed. History has not been kind to printer security. We haven't seen it exploited extensively. But as we know, more and more, if something can be done, it will be done. Google was kind enough to reward this group's findings with $3,133.70, which of course we know is LEE1E, essentially, in LEET.

Leo: Oh, it's upside down.

Steve: Exactly.

Leo: I was trying to think, it's like Euler's number? What is that number, 31337? Because 3 is E and 7 is L.

Steve: Right.

Leo: And there's one T?

Steve: So I guess, yeah, it's trying to be ELEET.

Leo: ELEET or something. Oh, I get it.

Steve: They're elite.

Leo: That's funny.

Steve: So, yes, just another little thing to worry about on a cold winter's day, whether anything has crawled into your printer and might not have your best interests at heart.

Leo: Wow. That's just...

Steve: Yeah.

Leo: Somebody in the chatroom said, and I completely concur, "I would hate to be a corporate IT guy these days." Just the stuff you'd have to keep track of.

Steve: Oh, yeah. And you're then - you've got the target on you. When something does happen, your boss is like, well, how did this happen? I thought you were in charge of this. I thought you were supposed to be some great security genius. It's like, uh, yeah, but our printer just got us. You know, who thought that was going to happen?

Leo: I'm not that big a genius. On the other hand, if you're the guy who's securing this stuff, and you do know it, you've got a job for life.

Steve: That's very true. Years ago, earlier in the podcast, people used to say, you know, I've been listening to this podcast for a while, and I'm thinking there might be some career opportunities here.

Leo: Yeah.

Steve: And it's like, I'm not going to have any hair left, and we're still going to be doing this podcast. Actually, I don't have that far to go.

Leo: I think it happened, Steve.

Steve: Maybe you, Leo, will not have any hair left.

Leo: I think it happened. Steve Gibson's at GRC.com. That's where SQRL lives. That's where SpinRite, the world's best hard drive recovery and maintenance utility lives. Lots of free stuff.

Steve: It doesn't give up.

Leo: It doesn't give up.

Steve: A SQRL that doesn't give up.

Leo: SQRL will never die. You'll also find Perfect Paper Passwords, a password generator if you want good, long, 64-character passwords, true random. You said it, but how do you know it's - you're not using pseudorandom number generator, I mean, for those passwords.

Steve: Yes, I am.

Leo: Oh, you are.

Steve: I am. Although it is running through a cryptographic cipher, with a secret key. So it's completely unpredictable and...

Leo: It's getting entangled.

Steve: Yes, and guaranteed not to repeat. So it cannot repeat, and it's unpredictable. That's the two things you want. It's never use the same one again, and unpredictable.

Leo: We've been meaning to do a hardware random number generator. What do you do, a diode? I can't remember what you use?

Steve: Yeah, yeah. You basically push a diode into breakdown, and the actual passage of electrons force back through the diode. The diode doesn't want to let them through. But if you put enough voltage on it, but limit the current, it creates noise. And it is quantum-level noise. So if you then take that and digitize it and then whiten it, because it can be - it's noisy, but it's not exactly uniform. So what you can do is you can take pairs and XOR them in order to whiten it and then end up with true hardware randomness.

Leo: I presume that LastPass and other passwords vaults, when they generate their random passwords, are doing something like you do, right, somehow? It's got to be a pseudorandom number generator. That's all computers can do.

Steve: Yeah. They could take some seed entropy from the server and then mix their own local entropy with it. And again, remember, as I said earlier, when you add entropy to entropy, you never get less. You only get more. And so it makes sense to sort of like start with some and then add some so that you can trust the source.

Leo: Got it. That's why I like this show. You learn so much. You can also find the podcast at GRC.com, audio and full transcriptions. Steve takes money out of his own pocket to pay to do that, and I think that's so great. I really appreciate your doing that. It gives you a searchable database. So if you're looking for any particular phrase or term, you can literally google, do site:GRC.com and google a term, and you'll find that podcast, and you can listen to it or read the transcript.

Steve: Yeah. And as this library gets increasingly deep...

Leo: Oh, it's amazing.

Steve: ...I'm seeing lots of interaction in Twitter of people saying, oh, just go over to GRC, and you'll find the podcast where we talked about that.

Leo: It's becoming kind of a compendium of everything you need to know about computing in one show. We have audio and video, if you want to watch Steve gesticulate. He's a fabulous gesticulator. And the video of that, the proof, is at TWiT.tv/sn. Or you can always watch live. We do the show Tuesday afternoon at 1:30 Pacific, 4:30 Eastern, 21:30 UTC. So you can watch it live. We are now on YouTube.com. The YouTube Live is at YouTube.com/twit. But you'll also find a link there to the show after the fact, so you can watch all the previous shows on video there at YouTube.

You can watch live.twit.tv/live or live.twit.tv or use one of the many apps. There's five Apple TV apps. There's Roku. There's a Roku app. We commissioned that from Craig Mullaney at ShiftKeySoftware. Lots of places you could watch live. If you are live, go in the chatroom, irc.twit.tv. If you want to watch after the fact, same thing. We've got podcasts, but there's also a podcatcher for your particular platform everywhere. Just search for Security Now! in that, and you'll find it and subscribe, and that way you get it every week. Thanks, Steve. We'll see you next week.

Steve: Okay, my friend. See you in February.