Hide Your RDP Now!

Description: This week we start off with something bad that we unfortunately saw coming. We then look at the changing security indication feedback in browsers; the challenge of keeping browsers compatible with important but non-standards-compliant websites; the failure and repair of incognito browsing mode; the possibility of a forthcoming "super incognito mode" for Firefox; a new super-fast TLS stack written in the Rust programming language; Microsoft's promised open source release of their voting machine election software; and yet another widely deployed, exposed, and exploitable Internet server. We have a quick bit of miscellany and some terrific SQRL news. Then we look at a recent and quite sobering report from Sophos about attacks on exposed RDP servers.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-724.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-724-lq.mp3

SHOW TEASE: It's time for Security Now! with Steve Gibson. I'm Jason Howell, filling in for Leo this week. Steve is going to talk about how Kazakhstan might actually be writing the man-in-the-middle playbook for authoritarian regimes, the Torification of Firefox, Microsoft's ElectionGuard, how incognito mode isn't very incognito when it comes to paid content sites, and Steve's RDP honeypot whitepaper. Fascinating stuff, coming up next on Security Now!.

JASON HOWELL: This is Security Now! with Steve Gibson, Episode 724, recorded Tuesday, July 23rd, 2019: Hide Your RDP Now!

It's time for Security Now!. I'm Jason Howell, filling in for Leo Laporte. That's right. He's gallivanting somewhere in the world. Joining me today is of course the man who makes this show entirely possible, Steve Gibson. Steve, how are you doing today?

Steve Gibson: Great. And great to be with you. I wasn't expecting you to be co-hosting, so I'm delighted. This is a treat. Is Leo off on business, or do we know? Someone must know what he's up to.

JASON: I should really know whether he's off on business, but I think it's pleasure.

Steve: Well, none of us know. But I was hearing about a rollercoaster. Maybe this is the - was this the rollercoaster trip?

ENGINEER: Yeah, yeah. It's Michael's 16th birthday, so they went to Venice with Michael and a bunch of his friends.

JASON: There we go. See, this is what happens when I take vacation last week, and they take vacation this week. I'm just completely lost and in the dark. I'm also in the dark because my computer is not turning on. So this will be really interesting today, Steve.

Steve: So your face will not be bottom lit from your screen, lighting you up.

JASON: No. Yes, exactly.

Steve: That's all right.

JASON: I won't be casting a shadow behind me, either. So it's all right.

Steve: So we have Episode 724 this week.

JASON: That's right.

Steve: And up until the last minute, I didn't have this title for the show, as you know, because you and I were talking for the last few hours as we were getting synchronized here. But a study that I read, a result of Sophos's setting up some RDP honeypots to get a sense for bad guys trying to log into Microsoft's Remote Desktop Protocol, the numbers were so compelling, and not in a good way, that I thought, okay, that has to be the topic for this week's podcast. And their conclusion was the same thing that I have recommended to our listeners while we've been talking about BlueKeep exploits and the problem of the fact that it's now possible to get into RDP servers without authenticating, the fact that there's really no safe way to have a remote desktop protocol exposed.

And so, for example, I use it like crazy. But there's no open port anywhere. And it's just not secure enough. It has had a troubled past. And so I titled this podcast "Hide Your RDP Now!" because one way or another you really need to have it hidden. But we'll get to that.

We're going to start off looking at something bad that has happened, which unfortunately we saw coming. I got a lot of tweets from our listeners over the last week saying, oh, my god, this is what you were worried was going to happen. And yes, looks like it is. Then we're going to look at the changing security indication feedback provided by browsers, in this case some changes coming to Firefox to follow what Chrome has done. The ongoing challenge of keeping browsers compatible when important but non-standards-compliant websites apparently are designed for one particular browser, but no one bothers to test them on other browsers. That turns out to be a problem.

Also we have the failure and then the subsequent repair of incognito browsing mode. The possibility, really interesting, of a forthcoming super-incognito mode for Firefox. A very fast new TLS stack written in the Rust programming language. Microsoft's promised open source release of their voting machine election software, which I'm very excited about. Yet another widely deployed, exposed, and exploitable Internet server, more than a million of them just waiting for people to hook up. And then we've got a bit of miscellany. Some terrific SQRL news, including my announcement of a trip into Europe to present SQRL to two OWASP groups.

And we're going to look at, as I said, we'll finish up looking at this very sobering report as a result of Sophos putting some honeypots up on the 'Net and just monitoring the bad guys. Oh, and actually this is relevant also to ransomware. It turns out this may be the way that they have been getting into these municipalities. So I think another great podcast for our listeners.

JASON: No, not "think." I know another great podcast for our listeners. And I was just actually checking in on your episode last week. Fifteen years, that's crazy. That's craziness.

Steve: Yeah.

JASON: You're like podcast king, you and Leo.

Steve: We're closing in. We're closing in. Next month on the 20th we will be beginning Year 15. So we're wrapping up Year 14 now.

JASON: And security has gotten not any less interesting or complex.

Steve: Yes. I remember when Leo suggested this to me, I remember thinking we're going to run out of stuff to talk about.

JASON: Right. Yeah, right. I think your job is secure. I think you're 100% secure in that. All right. Starting with the news that we have here, and it seems to be all about Kazakhstan. What's going on here?

Steve: Well, yes. So as I said, a whole bunch of our listeners tweeted this, and I got some private messages, too. Unfortunately, we sort of foresaw this. It was more of a - I guess it was more of a worry than a prediction. It's something that I've been hoping would not happen, but this may be just like the beginning of the break in the dike. We'll see.

If we look at the Picture of the Week on the first page of the show notes, if we put that on the screen, what we see on the left is the authentic Facebook certificate with the common name *.facebook.com, and then a long list of affiliated domains listed in its subject alternative name, the SAN field. So these are all of the different domains that that certificate is valid for. And that authentic certificate was signed, was issued by and signed by my favorite certificate authority, DigiCert. So that's the real McCoy.

Sadly, on the right we see a certificate that is in every way a clone copy of Facebook's authentic certificate containing the same *.facebook.com common name and the identical list of affiliated domains. But the common name of the signer of this fraudulent certificate reads "Security Certificate," which is not a valid CA.

JASON: Sounds legit to me.

Steve: That inspires confidence, doesn't it? And what we now know is that the government of Kazakhstan has begun officially requiring that its own CA root certificate be installed into the web browsers of its citizens. And of course we know why. It's so that it can do just what we saw. That certificate, that fraudulent Facebook.com certificate, was synthesized by Kazakhstan and signed by their root certificate. It of course would never be trusted. It's bogus. It's fraudulent. So it would never be trusted by anyone's browser anywhere, except if you had installed their root cert in your machine, which is what they are now beginning to require their citizens to do.

They say that this will allow them to increase the security and provide protection for the citizens of Kazakhstan. What we understand, of course, is that it allows essentially a man in the middle to intercept the traffic. We've talked about how enterprises often will do this. What was that - it's not BlueKeep, it's Blue something, I can't remember the name of it. There's a very often-deployed, you know, it's called a "middlebox." It's on the border of an enterprise. The employees in the enterprise install this certificate into their PCs. And actually it's done for them. It's typically done over the network. Anybody who

connects a machine up onto the network gets this certificate added to their machine. And what this allows the enterprise to do is to scan incoming and outgoing traffic for malware, for viruses, perhaps to do content control, whatever.

Well, it's one thing to do that in an enterprise, and an entirely different thing for a nation to do that to its citizens. The first indication of this, which is what some astute people saw and sent to me, was a Bugzilla report #1567114, which was titled "MITM [man in the middle] on all HTTPS traffic in Kazakhstan." This Bugzilla report reads: "Since today, all Internet providers in Kazakhstan started MITM on all encrypted HTTPS traffic. They asked end users to install government-issued certificate authority on all devices, in every browser." And unfortunately the links here are in either Kazakh or Russian, neither of which I read. So they're not very informational to English-only people.

But Matthew Hardeman quoted in the thread. So what this started was a dialogue that was held over on a thread, a discussion thread on Google. And his was, I think, one of the more insightful, balanced, you know, what are we going to do about this? And a bunch of Mozilla.com people were involved and copied on this. He wrote: "If the government of Kazakhstan requires interception of TLS as a condition of access, the real question being asked is whether or not Mozilla products" - because at this point this is over on the Firefox side - "Mozilla products will tolerate being used in these circumstances."

He says: "Your options" - he's writing to the Mozilla people. "Your options are to block the certificate, in which case Mozilla products simply become unusable to those subject to this interception, or not block the certificate." He says: "I certainly think that Mozilla should not distribute the MITM root nor do anything special to aid in its installation. I believe policy already makes clear that no included root, commercial or government, is allowed for use in MITM scenarios; and I believe that policy should be held firm. I do believe that, as it is manually installed rather than distributed as a default, that it should continue to override pinning policy. This is an accepted corporate use case today in certain managed environments," as we were just saying.

"The dynamic is quite different for an entire people under an entire government, but the result is the same. One has to choose whether to continue serving the user, despite the adverse and anti-privacy scenario, or if one simply won't have their products be any part of that. Much has been said," he writes, "about the TLS 1.3 design hoping to discourage use cases like this, but the reality is," he says, "what I always suspected: some enterprises or governments actually will spend the money to do full active MITM interception. Let's posit what might happen if Mozilla made their products intentionally break for this use case. Further, let's stipulate that every other major browser follows course, and they all blacklist this or any other nation-state interception certificate, even if manually installed."

And of course we know that's possible, just to take an aside here, because when the certificate comes to the browser, it chains up to a root. And Mozilla could check to see whether it's the "security certificate" from Kazakhstan, rather than a valid certificate like DigiCert's root. So it's possible from a technological standpoint to do this. It's trivial. And of course we've covered in years past where, when CAs have been caught doing things wrong, the browsers like Chrome and Firefox have blacklisted their roots. So, again, I mean, there are already blacklists in our browsers to prevent known bad roots from being used for any purpose at all. Kazakhstan could easily be added, like in an instant.

So he says: "Isn't the logical outcome" - that is, if all browsers blacklisted. "Isn't the logical outcome that nation-state forks one of the open-source browser projects, patches in their MITM certificate to undo the blacklisting?" He says: "I think that's exactly what would happen. The trouble is, there's no reason to expect that the fork will be maintained or updated as security issues are discovered and upstream patches are issued. We wind

up with an infrequent release cycle browser being used by all these users, who in turn get no privacy and get their machines rooted disproportionate to the global population."

He finishes: "I do definitely support a persistent UI indicator for MITM scenarios that emphasizes onscreen at all times that the session is being protected by a non-standard certificate and some sort of link to explain MITM and the risks." So he finishes, I mean, that's the end of his posting that I thought took a very good position on this.

So some background here. The Kazakh government first tried to have all its citizens install a root certificate three and a half years ago in December of 2015. And that sounded familiar, so I imagine it caught our attention, and we talked about it at the time. Back then, the government ruled that all Kazakh Internet users had to install the government's root certificate by January 1, 2016. However, that ruling was never implemented because the government was sued by several organizations, including ISPs, banks, and even some foreign governments who feared that this would result in weakened security of all Internet traffic, and weakened security for the businesses originating from the country. And of course they were, and still are, correct in that regard. At the time, in December 2015, the Kazakh government also applied to Mozilla to have its root included in Firefox by default, but Mozilla at the time declined.

So here we are now. And starting last Wednesday, July 17th, the Kazakhstan government has started intercepting all HTTPS traffic inside its borders. Local Internet service providers have been instructed by the local government to force their respective users into installing a government-issued certificate on all devices and in every browser. And as we know, once that's been installed, the certificate will allow local government agencies, well, pretty much anybody who has the matching root which would allow them to sign an on-the-fly-created certificate like that one we saw for Facebook. That was actually caught - that was caught on the wire. That was caught in use. So we know it's just not a theoretical question. I mean, there is a fraudulent Facebook certificate. And you have to imagine Facebook's not happy that their cert has been copied. Oh, well.

So in a statement posted on its website, the Kazakh Ministry of Digital Development, Innovation, and Aerospace said: "Only Internet users in Kazakhstan's capital of Nur-Sultan will have to install the certificate." However, users from all across the country reported being blocked from accessing the Internet until they installed the government's certificate. Some users also received SMS messages on their smartphones about having to install the certs, according to local media. Ministry officials said that the measure was "aimed at enhancing the protection of citizens, government bodies, and private companies from hacker attacks, Internet fraudsters, and other types of cyber threats." Yes, we're looking out for you by intercepting all of your otherwise secure traffic, which wouldn't have been subject to any of those problems before.

Now, it's true, as we know, this sort of interception could be used to benignly filter traffic for malware, just as corporations do. But unfortunately it's equally possible to scan and surveil traffic for whatever other content the government or other MITM agency might choose. So as a consequence of this, at the moment, Google, Microsoft, and Mozilla are gathered to discuss a plan of action about dealing with websites that have been reencrypted by the Kazakh government's root certificate. It's going to be very interesting to see how this develops. And what seems most likely is that, as the person who I quoted suggested, that the browsers and other devices will adopt some form of persistent visible warning, you know, like all the chrome around the edges of the browser might turn scarlet or something.

JASON: Or flash at you.

Steve: Yeah, I mean, you know...

JASON: For the entire time that you're using it.

Steve: That's right. And it's true. Since the industry's best browsers are now all open source, private labeling a Kazakhstani browser containing only Kazakhstan's root certificate would not be an insurmountable task - although, as he also noted, maintaining it and keeping it current would be a big job. Which means it probably wouldn't happen. And people would have a going-out-of-date browser that is the only one that worked in Kazakhstan, which really does not really seem to be serving the best interests of its users.

So, you know, most users will have no idea what this is about. They're told to do something in order to, you know, their government, oh, the government needs me to install something. Okay. And so what choice do they have? I'm sure the media will give them a sense for what's happening. Anyway, so it's either lose your privacy or lose the Internet. You choose. And of course it's also likely that other repressive and authoritarian governments are probably watching this to see what happens because they'd all like to do it, too.

Now, in some later follow-up reporting, the water was a bit muddied, although based on the fact that people outside of the capital were reporting they could not access the Internet at all until they added the certificate, it really does sound like what this official cited. Caleb Chen, writing for the PrivateInternetAccess blog, noted, he said: "A Kazakh official clarified on the 19th that the installation of the certificate was voluntary and not a prerequisite to accessing the Internet."

Now, as I read that again, I'm thinking, well, maybe they meant, like, email or something that is not using HTTPS and browser certs. So maybe this guy is sort of trying to say, oh, yeah, you could still use the Internet; but, okay, yeah, but not your browser, which is for many people the Internet, just as "the Google" is the way you find the Internet. Anyway, Caleb provided a link to the page, and I have it in the show notes for what good it does anyone. Maybe if you read Kazakh or Russian, and I'm not even sure what language it was written in.

JASON: Google will translate it for you, by the way.

Steve: It's Greek to me. And that translation's getting better, too, isn't it.

JASON: It's getting really good. I'm really impressed.

Steve: Yeah, I've not used it for a while. So anyway, so what do tech-savvy Kazakhs think about this? The Mozilla bug thread contains a posting from one. He says: "I am a citizen of Kazakhstan. If Mozilla/Google Chrome developers see this message, I kindly ask you to consider blocking the above-mentioned certificate and any access to your browsers for the certificate holders. If this certificate didn't pass web trust audit, it can be the same as presented in 2016," meaning same outcome. "So blocking it from the major world browsers is the only chance for Kazakhs to avoid MITM attacks and keep at least some privacy rights, meaning that, if blocked/blacklisted, the government will have to call back the certificate, as was done in 2016."

He finishes: "If the certificate is not blacklisted, but only the visual message will pop up warning users about untrusted certificate, it will not help, since majority of citizens, especially elderly ones," he writes, "simply will not pay enough attention to such a message."

So of course this has caused an industry-wide kerfuffle. Our illustrious crypto professor Matthew Green weighed in via Twitter. He tweeted, Matthew tweeted: "This is the future all those cypherpunks warned us about. Let's hope the West sees the danger and doesn't

look on with admiration." So, yeah, that has happened. And we'll see how long it lasts. There was a big pushback. And as our listeners know, I have worried that at some point we might see, we in the U.S. might see our ISPs that are the perfect place to intercept and filter our HTTP traffic "on our behalf."

JASON: To protect us.

Steve: Yes. Your ISP is the entity from whom you purchase your bandwidth. So it would be logical for them to say, okay, we're upgrading your service. You need to install this certificate in your browser in order to continue accessing websites on the Internet through us. Boy.

JASON: Not a bug. It's a feature.

Steve: Hasn't happened yet. And boy, I'll be surprised if it does here. But we'll see.

JASON: Yeah, I mean, that last comment from Matthew Green really feels like a 50/50 chance it could go either side of what he's saying there, either seeing the danger or looking on with admiration. You just really don't know. That's kind of where we are right now.

Steve: Well, and as we've said, our government is unhappy that it's unable to see into our browser sessions. And so I don't know.

JASON: Yeah.

Steve: Speaking of indicators and browser security and privacy, starting in October of this year with Firefox 70 - so that's two from now, we're currently on 68 - Mozilla will be prominently marking all HTTP, that is, non-HTTPS pages as not secure, the way Chrome does now. What Mozilla has been doing with Firefox so far was only choosing to show not secure indicators on HTTP pages, where it probably mattered more, meaning pages which accepted some user-provided content containing forms or login fields. However, today the percentage of sites serving HTTPS has now surpassed 80%. I mean, it's just what you do. So their thinking is that users don't need good news for what is rapidly becoming the default, that is, being secure, but rather a warning when, for some reason, a site is not secured by HTTPS because now that's the exception.

Firefox developer Johann Hofmann wrote: "In desktop Firefox 70, we intend to show an icon in the identity block" - that's the left-hand side of the URL bar which is used to display security/privacy information, and that's actually getting kind of crowded over there. I had to use it this morning. I wanted to check out - I was monitoring MacBreak Weekly, and they were talking about the new "Picard" trailer for the "Picard" series coming out next year on CBS All Access. Leo and I had spoken of it. I'd seen the very first teaser, and I knew that at Comic-Con they had released the first long trailer.

Anyway, so I wanted a high-resolution one, and I figured that CBS All Access would be the place to get it. So I went there, and I am a subscriber, and so I logged in. But it wouldn't show anything, saying that I was filtering the traffic, and their sponsors were trying to show me something that I wasn't letting them show. So I thought, huh. And so I disabled uBlock Origin and then refreshed the page, and it still wasn't happy. And I thought, okay. Then I saw, okay, maybe something's still filtering stuff. So I switched to the in-private browsing, still wouldn't display.

And then I remembered that I had turned Firefox's anti-tracking all the way up to 12. So sure enough, one of the little icons over there, which is what made me think of it, was there. I clicked on it, and I made an exception to CBS.com, and then I was able to

happily play the 2:08 long "Picard" trailer, which I do recommend to all of our sci-fi interested listeners. It does, I mean it's annoying that it's not part of a broader subscription like Netflix or something. And CBS All Access, I think it's $10 a month, so it's not cheap either. But the upcoming series does look wonderful. And how fun to have Jean-Luc back.

Anyway, the point is, Firefox will be adding yet another thingy over in the URL bar to show people when they need to do something. Now, it has had configurable behavior flags which were viewable and could be set, accessible through its about:config page, since the end of 2017. Those flags are still present in the current stable version of Firefox. And anybody who is interested can preview the forthcoming indication, if you go to about:config, and then in the search bar - because as we always say there's just a bazillion things in that about:config section. If you search for "security.insecure_connection," that's probably enough.

That'll bring up four different options. One is to show a broken lock on HTTP sites. One is to show "not secure" text on HTTP sites. The other is show a broken lock on HTTP sites in private browsing. And the fourth, show the "not secure" text on HTTP sites in private browsing. And you just did it. And it looks like there's more than four. So there must have been a couple other things that are within that same search term. Or maybe underscore connection. If you add underscore connection, that might weed it down further.

JASON: Right, yeah, looks like that would cut it down to four.

Steve: Okay. And as I was writing this, I thought, okay, but it can now be difficult to find a non-HTTPS site for testing. Well, it happens I have one. And I used it the other day because I was curious about this. Although my new GRC link shortening service of course supports HTTPS, and it always redirects users to secure pages, I didn't see when I was setting it up any reason not to allow it to accept an incoming link over HTTP. Browsers are still trying that first. At some point browsers, if you don't give it an explicit https://, someday they'll just assume you mean "s" and try that first. Today they're still just going to HTTP. So that means you have to have the - they try to bring up the page. You need to be bound to that port. You need to then redirect them over to the "s." And so blah blah blah. I didn't see any reason to do that.

So for our listeners, anyone who's interested about how, if you want to check any browser to see how it looks on an HTTP site, though they are becoming increasingly rare, you can go to http://grc.sc and just bring up the root page, you know, .sc/ to bring up the root. I have just a little placeholder text there. But it will leave you. It won't redirect you to HTTPS. It'll leave you there. And you sure enough can see how Chrome and Firefox and what other browser you might be using and with configuration settings and things, how that looks in that mode. So anyway, just a little tip since it's becoming hard to find non-HTTPS sites these days.

And speaking of Firefox's about:config page, Mozilla's Dan Callahan, writing about the changes in the latest, that is, our current Firefox 68, he was explaining about some of the subtleties of browser behavior which were required to allow Firefox to operate on websites that were apparently written to expect some specific quirks of some non-Firefox browsers. He wrote: "Keeping the web open is hard work. Sometimes browsers disagree on how to interpret web standards. Other times, browsers implement and ship their own ideas without going through the standards process."

He says: "Even worse, some developers intentionally block certain browsers from their sites, regardless of whether" - I can't imagine doing that. What is wrong with these people? Anyway, "regardless of whether or not those browsers would have worked." That's got to be - but he just wrote it, so I guess, I mean, it sounds like the dark ages.

Anyway, "At Mozilla," he says, "we call these 'web compatibility' problems, or 'webcompat' for short. Each release of Firefox contains fixes for webcompat issues. For example, Firefox 68 implements," and then he says, "Internet Explorer's addRule and removeRule CSS methods." Which, okay, so I guess IE just - IE? Wow. IE has the ability to have JavaScript add or remove CSS methods. Okay. And then also Safari has a CSS property, -webkit-line-clamp.

And he says: "In the latter case, even with a standard line-clamp property in the works, we have to support the -webkit version to ensure that existing sites work in Firefox." And he says: "Unfortunately, not all webcompat issues are as simple as implementing non-standard APIs from other browsers. Some problems can only be fixed by modifying how Firefox works on a specific site, or even telling Firefox to pretend to be something else," you know, via a lie being told in the user-agent header, "in order to evade browser sniffing. We deliver these targeted fixes as part of the webcompat system add-on that's bundled with Firefox."

And the reason I'm telling everybody about this is I never knew this was in there. And now they've just surfaced it on the UI, and it's very cool. He says: "This makes it easier to update our webcompat interventions as sites change, without needing to bake those fixes directly into Firefox itself. As of Firefox 68" - that is, the one that we all have now - "you can view and disable" - although why would you - "these interventions by visiting about:compat." So that's my big tip, which is what this is about. We all know about about:config. We're talking about it all the time. About:compat, C-O-M-P-A-T. And sure enough, I never knew about that. It's only just recent been created. So I put that into my Firefox, and up comes kind of an interesting list of sites you've kind of heard of. Some are important, some less so. But they are sites where Firefox checks its compatibility, this webcompat feature, and alters its behavior per site in order to work right. So kind of cool.

He says: "Our first preference is always to help developers ensure their sites work on all modern browsers, but we can only address the problems that we're aware of. If you run into a web compatibility issue, please report it to webcompat.com." So they also grabbed that domain for their own efforts. So anyway, nice piece of work to keep Firefox compatible. And, wow, you know, it's unfortunate even today we still don't have a single standard. We still have disparate things that random browsers just do because they want to. Crazy.

JASON: And they all seem to at some point, hopefully, lift each other up onto the same plane. But, yeah.

Steve: Well, one of the nice things with Edge, the fact that Microsoft gave up their own Edge engine, and they're now going to be adopting the Chromium engine in Edge for Windows, automatically one fewer browser to be incompatible because, you know, it'll be an Edge browser with a Microsoft wrapper.

JASON: Right, right. This next story, it speaks to me a little more than some of the other stuff that we've talked about today, only because in the preparation that I do for the shows that we do on the network, we've got subscriptions to a lot of the sites that we pull news from. But we don't, I mean, obviously we can't buy a subscription to every single site. And every once in a while you run into that free article limit, and dang it. I want to read this. I want to invite that person on to talk about the article, and I can do nothing. Well, I guess I could sign on, but anyways.

Steve: So what you're talking about is that we've probably all encountered teaser paywall websites that allow a limited number of articles to be viewed per month by non-subscribers. And this sort of feels like a workable compromise between fully open and

fully closed sites. They're clearly hoping that you'll, you know, they're wanting you to see all their goodness rather than never be able to see any. But obviously they're hoping to upsell you into establishing a subscription. So they're hoping that you'll find value there, and then be annoyed when that one article you really want to read fades out into that notice that the site requires a monthly fee for you to keep reading. I get hit by that all the time, just like you, Jason, when you're putting the shows together.

And this, of course, begs the question: How are they counting the number of pages you've visited? The answer could be some form of fingerprinting. But we know that browser fingerprints, while containing some entropy, are not unique. Many other browser users will coincidentally have a browser with the same fingerprint since it's just based on a bunch of characteristics that are visible from script on your browser. And these sites are not attempting to block rocket scientists, after all, from having access. They're only trying to put up a barrier to casual users, which is to say it's not like it's impossible to get around this.

So the way they're clearly doing it is by the universally available first-party domain cookies, which everybody has enabled in order to effectively use the Internet. So it wasn't long before people who want to read that one additional article, but didn't want to join up, discovered that switching to their favorite browser's incognito or private browsing mode, whatever it's called on your browser, made that possible, since private browsing deliberately does not record first-party domain cookies or, for that matter, anything else. A user of an incognito browser is suddenly unknown, and always starts out with a zeroed prior articles counter, since they appear as an unknown, uncookified visitor.

And of course in this cat-and-mouse world, just as it didn't take long for the users of private browsing for this trick to become widespread, either to be discovered independently or spread by word of mouth, neither did it take the web developers then long to figure out a way, at least in the case of the Internet's number one web browser, Chrome, how to detect when somebody was visiting their paywalled site incognito. And then, for example, if you visit any article on the Washington Post's website using Chrome's incognito mode, you'll be greeted with the message: "We noticed you're browsing in private mode. Private browsing is permitted exclusively for our subscribers."

Wait. What? If you're a subscriber, then they know who you are, and that's not very private. But they said, and their note continues: "Turn off private browsing to keep reading this story, or subscribe to use this feature, plus get unlimited digital access." But wait. What? The idea of being identified, even as someone who is visiting incognito, sort of puts the lie to the whole incognito thing; right?

JASON: Indeed.

Steve: It's like, wait a second, you're not supposed to know anything about me, including, and perhaps even importantly, that I have deliberately chosen not to have you know anything about me, including that. So this occurred to the engineers at, as Leo used the term last week, "the Goog," which is now...

JASON: I love it. It's an endearing term, "the Goog."

Steve: The Goog, yes. This occurred to the engineers at the Goog. So they decided to do something about it. It turns out that Chrome's incognito mode disables its file system API, as you would want it to, as part of Chrome's effort to prevent the user's activities from leaving any lasting traces in the system. Websites figured out that this could be easily checked with a bit of JavaScript running on the page. So that's what generates that "Please disable your incognito mode to receive a limited amount of free stuff" message. Consequently, at the end of this month, actually it'll be July 30th, one week

from today, we're going to be getting Chrome 76, with a file system API that no longer gives away the fact that the browser is in incognito mode, or the browser's incognition.

And just in case publishers then go searching for some other method to detect private browsing, since Google really does wish to avoid having its incognito visitors flagged, the Goog has stated that "Chrome will likewise work to remedy any other current or future means of incognito mode detection." So be forewarned, you web developers out there. The Goog is going to work to protect its users' privacy. And now we all know, I didn't realize it before, but that's going to come in handy now, that I'll be able to use in-private mode or incognito mode or whatever it's called to avoid those notices. So thank you, Goog.

JASON: Thank you, the Goog.

Steve: And taking it up another notch, and this is very exciting, Firefox may be - and it's premature, but worth discussing - may be getting super-private browsing in the not-too-distant future a la Tor. Due to the fact that the Tor browser is a descendant of Firefox, there have been various projects and grants awarded over time to look into the possibility of adding a Tor mode to Firefox. This is exciting because setting up a Tor session takes some doing. And also because wrapping all outbound traffic in a multilayered, successively encrypted onion, and then having that traffic bounce three or four times around Tor nodes while each successive layer of the onion is stripped and decrypted and then forwarded on to the next Tor node, means that surfing the web through Tor is not for the impatient.

But the idea here is that Firefox users would be able to just flip a switch, the way we currently do when we turn on private browsing, in order to get the benefit of Tor on an as-needed basis. It would allow Firefox users to just drop the cone of silence over their online activities when they need some real, I mean like some robust anonymity, and then just as easily lift the cone to their regular high-speed browsing.

So it turns out that during a meeting of the core team, developers, volunteers, and invited guests in Stockholm, they gathered to discuss plans, milestones, deadlines, and other important matters. The idea of a Tor mode add-on for Firefox was proposed and is being considered. I have a link to it, for anyone who's interested in more details, in the show notes. What was said on that page is there is an idea to, in the future, have Firefox use Tor in private browsing mode or in a new extra-private mode. They said that will take a lot of engineering work and buy-in, that is to say, meaning to innately, natively embed Tor into Firefox. So they're kind of wanting not to do that.

So they said, to help smooth the path, there is a proposal for a Tor mode add-on. This would not be packaged with the browser by default, but would be something that users could download from addons.mozilla.org to give them a Tor mode button or similar. It would allow users to experience what an eventual full integration with Tor would look like. It could also help gauge interest by counting downloads and usage and so forth.

Someone whose handle is "acat," his name is Alex Catarineu, has demonstrated how to compile Tor to WASM. So that's been done, you know, web assem. This would allow packaging all the necessary Tor code inside the add-on itself, without a dependency on an external binary. The add-on would still need to be privileged. A privileged add-on is one with elevated privileges compared to a standard web extension. It can call XPCOM functions, for example, which are otherwise privileged. A privileged add-on needs to be signed by Mozilla, but the idea for this proposal is to have it produced and distributed by Mozilla anyway, so that's not a problem.

The add-on would configure the browser to use Tor as a proxy, as well as setting various prefs to prevent proxy bypasses and resist fingerprint and so forth, much like those that

are currently being set by the Tor browser, which is based on Firefox. Discussion of visual options for UI, under that topic they said: "Clicking the Tor mode button would probably open a new window that uses a dedicated profile. This is because some of the prefs that the add-on has to set are global to a profile, not to a window or tab."

So that is to say, it could not be a Tor mode tab on an otherwise non-Tor mode browser window. It would just have to give you a new window. And then they asked what to do about HTTP. The feeling is that it's dangerous to pass unauthenticated HTTP through exit nodes. Packaging NoScript does not provide the best experience, either. The easiest solution is to enforce and require HTTPS when in Tor mode.

So this is well in advance of any timeline or any kind of a release number target. But it's neat that the Tor gurus are thinking along these lines. As for myself, beyond experimenting with Tor just for the experience, like to play with it in order to talk about it for the podcast, I've never used it seriously because I don't have a big need for persistent anonymity when I'm on the 'Net. But I for one would add a Tor add-on to my browser if it made it just as simple as clicking a button for those few and brief times when being truly stealthy would be useful, and when I could put up with a much slower browsing experience because it is a lot slower. But still, very cool that that kind of idea would be forthcoming.

Rust has been in the news recently, the language, because Microsoft recently announced that they were going to be seriously considering Rust as the implementation, the systems implementation language for their projects moving forward, which is to say moving away from C, C++, and even C#, which is their own C. And along those lines, Rust - and it's hard to pronounce this because we need two T's. It's Rust-TLS, but it's Rustls, so Rustls, I guess.

JASON: That seems about right for me, Rustls.

Steve: Yeah, Rustls.

JASON: Or Rustls.

Steve: That's, yes, Rustls have ridges.

JASON: Right.

Steve: So it turns out that somebody wrote a TLS stack in Rustls and it outperforms SSL in nearly every way. It is a tiny, currently, well, it's tiny and a currently relatively unknown TLS library written in Rust. Benchmarks were performed by its author, Joseph Pirr-Bixton - or, I'm sorry, Birr. I got the "P" and the "B" backwards. Joseph Birr-Pixton, who is the author of the Rustls library. Joseph's finding showed that Rustls - I love saying that, Rustls - was 10% faster than OpenSSL when setting up and negotiating a new server connection, and between 20 and 40% faster when on the other end receiving or initiating setting up a client connection.

And of course these days most TLS traffic relies on resuming previously negotiated handshakes, that is, if the endpoints agree on the material that they had previously established, then they're able to more quickly - they're able to cut out some of the handshake traffic back and forth and just say let's continue using the existing cryptographic material that we had previously exchanged. So that is resuming existing connections. And there it's between 30 and 70% quicker to resume a client connection. Moreover, Rustls fared better in bulk data transfer, as well. Joseph's measurements show that Rustls could send data 15% faster than OpenSSL and receive it 5% faster. And just to put a cherry on top of all that good news, Rustls uses only half the memory consumed by OpenSSL.

Now, we've spoken of OpenSSL frequently because it's been a core of the Internet. It is old, and it's becoming creaky. And we're beginning to see more and more alternatives appearing. But it has been the workhorse. It's where all new SSL-related ideas are first developed and proven, and then over time they evolve. So it's evolved SSL all along the way, acquiring all of SSL's additional features and being the test bed for them. And then of course it was used to make the straddle from SSL to TLS, and then to evolve TLS. And we all know that anyone using any brand new TLS stack should initially do so with an extra supply of suspenders because TLS is a complex protocol, and you don't want to find your pants down around your ankles. Which is to say it's neat that there is a brand new fast TLS stack written in Rust. But I would be cautious because, you know, it's just it's hard to do this right the first time.

That said, Rust is probably well suited to this sort of application. Unlike C and C++, Rust has security designed into the language itself to avoid memory-related security bugs. Microsoft did a study, and they showed that roughly 70% of all the problems they find in their own code or, embarrassingly, other people find in their own code, 70% are memory allocation, memory management, buffer overrun, buffer underrun type bugs. And now it's looking like Rust is generally able to outperform code written in C and C++.

Rust was initially developed by Mozilla. Firefox since 2016 has had Rust in it. It's been Rusty, and getting more so. And I remember that it was ridiculed at first, wasn't really taken very seriously. But as it's proven itself, it's really becoming a serious language. I think, when I was doing some background research, I think all of the Stack Overflow surveys since 2016, so 2016, '17, '18, and even '19 have all put it as number one most favorite language. It has a clear, clean syntax. It's easy to learn. I mean, people are loving it. So I think it's a strongly typed language. So it's got a lot of the features that programmers are liking to help prevent them from making errors.

Firefox and Brave browsers are relying on Rust components, and large companies are adopting Rust. Cloudflare, Dropbox, Yelp, and NPM have adopted it for their production systems. The Tor Project is experimenting with Rust. And even Facebook's recently launched Libra cryptocurrency will be using Rust. And as I mentioned, last but not least, Microsoft just recently announced their plan to use it. So, oh, yeah, here I had in my notes, it's come out on top as the most popular programming language in Stack Overflow's developer survey the past five years, although there's only four years - 2016, 2017, 2018, 2019. So maybe it's going to be the same way in 2020, who knows.

JASON: Yeah. Yeah, look at the positive.

Steve: Positive outlook. For anyone who's interested, it's Rust-lang.org, R-U-S-T hyphen L-A-N-G dot org. And on that site, which is of course pro-Rust, they say of performance: "Rust is blazingly fast and memory efficient. With no runtime or garbage collector, it can power performance-critical services, run on embedded devices, and easily integrate with other languages." Of reliability: "Rust's rich type system and ownership model guarantee memory safety and thread safety, and enable you to eliminate many classes of bugs at compile time." And of productivity: "Rust has great documentation, a friendly compiler with useful error messages, and topnotch tooling: an integrated package manager and build tool, smart multi-editor support with auto-completion and type inspections, an auto-formatter, and more." So anyway, it does look like it's something that's worth looking at.

So I wanted to, first of all, put Joseph's Rustls library on everyone's radar in case there might be some interest and need, and also because computer languages are fun. As I've mentioned before, I plan to accelerate my own development of SpinRite 7, not 6, not the 6-dot releases, but the 7, because I plan to add a large array of features. Whereas 6.1, .2, .3 and so forth will be performance-oriented, support-broadening work. What I plan to do is to develop the low-level driver code for the SpinRite 6.x series, but then to

implement the next UI, and the technology underlying the UI, in something that's faster for me to prototype and experiment and work with. I had been thinking Python for that, but I'll definitely take a look at Rust. And I know that Leo would be saying, "Steve, look at LISP. Look at LISP." But no. I'm not writing SpinRite 7 in LISP. But maybe Rust.

I mentioned Microsoft and election security. I'm excited about this. Last week during the Aspen Security Forum Microsoft announced ElectionGuard. Well, they announced and demonstrated. They had some kiosks running. It's a new open source technology which can be used to secure modern electronic voting machines. And thank god it is not the beginning of Microsoft balloting machines based on Windows 10. No, we're not going to have that. Microsoft stated that it had no plans to release commercial voting machines. Actually, this came out of Microsoft Research, which is a different group. Instead, they said they plan to release the ElectionGuard software on GitHub under an open source license later this year.

They described the technology behind ElectionGuard as being relatively simple and centering around a few core principles, which is exactly the right approach. And what's interesting is that somehow we've covered this before. So maybe we talked about it in a very pre-release mode. But now it's demonstrable. So here's what we know. Voters who vote with the machine receive a tracking code from the act of voting. The tracking code can then be used on an election website to verify that their vote has been counted, and that that vote was not altered.

But the tracking code reveals nothing about the voting, so it will not allow third parties to see who voted for whom. It uses a form of homomorphic encryption which was developed in-house at Microsoft by their cryptography team. Homomorphic encryption, difficult to pronounce, is a very cool technology that allows the counting of votes while keeping the votes themselves encrypted. The ElectionGuard SDK also allows third-party verifier apps to independently check if encrypted votes have been counted properly and not altered. So there is no single point of authority. It's inherently a distributable and distributed technology.

The verifier apps were created for voting officials, the media, or any third party interested in the voting process. So basically it kind of makes it open and auditable, inherently. ElectionGuard Machines can also produce paper ballots as a printed record of their vote, which voters can place inside voting boxes like old-fashioned votes. And it supports voting through accessibility hardware such as Microsoft's Surface, or they give the example of the Xbox Adaptive Controller.

So it's already attracted interest, as it should have because it's the right solution. Microsoft hopes voting machine makers will use its new ElectionGuard software for their own products. And according to Microsoft, the SDK has been warmly welcomed by some voting machine vendors already. Tom Burt, who is Microsoft's corporate vice president in charge of customer security and trust, said: "We previously announced that we have partnerships with suppliers that build and sell more than half of the voting machines used in the United States. Today we're excited to announce that we're also now partnering with Smartmatic and Clear Ballot, two of the leading voting technology vendors; and Dominion Voting Systems is actively exploring the inclusion of ElectionGuard in their offerings."

And from my standpoint this is exactly the correct direction for our voting systems. As I said before, it's fine for the likes of Diebold to manufacture and sell the hardware. But the way their machines work and what they do cannot be allowed to remain proprietary secrets. That's just ridiculous, the idea that voting machines are proprietary, and for there to be a "just trust us" model. As we know, they are famously riddled with vulnerabilities. And they don't begin to provide this next-generation style of really leverageable crypto that the ElectionGuard technology offers.

So the entire notion that some random hardware manufacturer has some kind of secret sauce to protect our votes is utter nonsense. And it will take someone with a solid reputation, just like Microsoft Research, to offer an answer at no cost to the hardware vendors. And it being open inherently means all the academicians can attack it and see if they can find any problems. So I just say a big yay to this. It's really good news.

JASON: Yeah, makes a lot of sense.

Steve: And of course now what we need, once the new machines exist, we need the old ones to be taken out of service because I've seen this massive...

JASON: Right. And that'll happen really fast.

Steve: Yeah. Well, it'll happen, especially if governments are required by law to have machines that support ElectionGuard technology. That would be perfect.

JASON: Absolutely.

Steve: And we always, you know, what could almost be a weekly installment of Security Now!, we have yet another remotely exploitable vulnerability of a widely used, high-profile, and easily discoverable Internet server. Yes, more than one million currently online instances of the popular open source ProFTPD server, which is present in distributions of Debian, SUSE, Ubuntu, and many other Linux and Unix-like systems, is open to remote compromise. And that is to say has always been so, including remote code execution as a result of an arbitrary file copy vulnerability.

What's on the screen right now is the result of a Shodan search for ProFTPD daemons. This shows nearly 203,000 of them in the U.S., 182,000 in Germany, almost 90,000 in France, 55,000 in the Russian federation, 48,000 in the Netherlands, 47,000 in Japan, 41,000 in the U.K., 35,000 in Spain, almost 28,000 in Hong Kong, and 24,000 in Italy, totaling more than a million. All ProFTPD versions up to and including the latest, 1.3.6, which was released back in 2017, are currently vulnerable unless their default configuration has been modified, and we know what that means. They're all currently vulnerable.

The problem is ProFTPD has a mod_copy module which implements additional commands which any anonymous FTP client can issue. The ProFTPD site explains: "The mod_copy module implements SITE CPFR and SITE CPTO" - that's copy from, copy to - "commands which can be used to copy files/directories from one place to another on the server without having to transfer the data to the client and back." Okay. So the ProFTPD developers, they said, hey, let's enhance FTP. Rather than requiring you to download a file and then upload it somewhere else, let's just allow you to give some commands to cause the FTP server to perform essentially an intra-site file or directory copy.

So they are anonymous. That is, the SITE CPFR and SITE CPTO can be used by an anonymous, non-authenticated FTP client which connects to any ProFTPD server and can simply arrange to copy any file that's there to somewhere else. Okay. And of course anybody who's used FTP can immediately begin to see the problems. There is a bug report on the ProFTPD.org site which says a ProFTPD user reported the following via email: "The mod_copy module's custom SITE CPFR and SITE CPTO commands do not honor the <Limit READ> and <Limit WRITE> configurations as expected. To reproduce, just enable the anonymous user example that is preconfigured in the Debian default proftpd.conf." And then they show us a sort of a standard Linux-style, Unix-style configuration where you have an anonymous FTP user block opened, and then you define the user, the group, the user alias, max clients, the login message to display and so forth.

And then within that anonymous definition you have <Directory *>, meaning for all directories, <Limit WRITE>. And then in there is DenyAll. Then you close the limit, then you close the directory, and you close the anonymous specification. So that's your standard. So that's saying do not allow writing to any directories. Then in this ongoing bug report they say issue the command ftp space and then whatever, you know, proftpdtest.domain.org as an example, meaning just connect to the server from any client. Login as anonymous, and then you issue the site space cpfr. And then in the example they gave, space welcome.msg, the welcome message. Then you say site cpto malicious.php, which has the effect of copying whatever was in the welcome.msg file into malicious.php.

And of course that example is not going to be particularly useful, but they're not going to want to put an actively useful malicious example in their bug report. This just demonstrates the problem. But as many FTP systems are set up, and many people know, it's often the case that FTP servers will designate, for example, an uploads directory where anonymous people can submit stuff. They can upload things to the uploads directory. They can't put them anywhere else, and they can't do anything with them. But it's like, you know, upload the file to the uploads directory. Then you contact the site admin and say, hey, I just sent you a sample of something, might be malware, might be a virus, might be whatever. Go do with it whatever you want to.

The point is now the bad guys upload something to the uploads directory. Then they issue these two commands to copy the thing they uploaded that they should have absolutely no other access to except uploading it to that directory. They now copy it wherever they want to on the server. So it's very clear that this is a very bad problem. And if any of our listeners do have a ProFTPD server up and running, you want to immediately, and that's the remediation, immediately disable that mod_copy module. That mod_copy module was the add-on which adds these two commands that no one's probably using anyway.

So it's another example of this should never have been turned on by default and left on by default. One wonders why it was even written as an extension, but there it is. So make sure you're not using it. That would be - you're going to be much happier not to have people basically replacing files on your server, anywhere on your server at their whim, based on FTP. Yikes.

I have a note from Patrick Delahanty, who is a web engineer.

JASON: Who? Who?

Steve: People have probably heard his name.

JASON: Dela-what-hanty?

Steve: Uh-huh.

JASON: Oh, okay, TWiT. Oh, so he works here. He's one of the hundreds of people walking past me in the hall on the way to the studio.

Steve: One of those many people who keeps the lights on, yes.

JASON: And he's about to come in here and glare. Okay, so he's glaring at me right now. I knew that was going to - literally, his office is right on the other side of the studio door. So hi.

Steve: I got email from Patrick this morning. He said: "Hi, Steve. I've been making some simple Apple TV apps for some TWiT shows that have evergreen content people like to

watch even after the content has been rotated out of our podcast feeds. I've released a Security Now! app for Apple TV that uses the TWiT API to access content and lets people watch or listen to every episode" - yes, all 724 - "going all the way back to the beginning."

JASON: Nice.

Steve: So for everyone who's listening: http://twit.to/sntv for Security Now! television. And Patrick, thank you very much for doing that and for letting me know so I can let all of our listeners know.

JASON: Looks really good. Yeah, looks awesome.

Steve: Very cool. So I have some SQRL news.

JASON: Nice.

Steve: Yeah. Scott White has sqrl-ssp working, which is a very nice, pluggable implementation of the SQRL service provider API, which he has written in Go. And I know from interacting with him over in the SQRL forums that he developed it specifically to be very horizontally scalable. My own implementation is a sort of a single-server approach. And so we've been going back and forth. And what he's got written in Go up on GitHub is highly scalable for much bigger installations.

Paul Farrer has a reference implementation of the SSP API in C, which he closely translated from mine, which I provided to him. I gave him the source, which is in assembly language, because Paul's also comfortable with assembler. And so it's a feature-complete implementation and supports several different TLS stacks.

Daniel Persson, who created the Android client and the WordPress plugin, both which then evolved into very effective team efforts, is now working on a sqrl-libpam module, PAM of course being a Pluggable Authentication Module for Linux. So that's on the way.

Adam Lenda, working with PHP and SQRL, is working to add SQRL support to the PHP Symfony application development framework. So that's in the works. Jurgen Haas has SQRL for Drupal 8 coming. And, meanwhile, the Android, iOS, and web browser extensions are rapidly nearing completion. Those are all clients.

I finished the second of the four SQRL documents to fully specify SQRL. As we know, the first one was "SQRL Explained," and the next one is titled "SQRL Operating Details." Before making it widely available, and it's not quite yet, I notified those in GRC's newsgroup, the NNTP newsgroup, who have passed a very fine-tooth comb through it, finding many typos, and also noting grammatical mistakes. Thank you very much, all. So I will be revisiting the document to incorporate all of their suggestions as soon as I'm through with the podcast. And I'll post it widely as soon as that's ready, so I think in a day or two.

And I'm now at work on the third document, "SQRL Cryptography," which will detail all of the crypto glue that holds SQRL together. And then the final document will be "SQRL on the Wire," which will document the communications protocol down at the character level. When these are finished, we'll have both a top-down and a bottom-up view of SQRL. So I am very excited to get the documentation wrapped up.

And last, but certainly not least, I'm very excited to have been invited to present SQRL in late September to two OWASP groups which will be meeting in Dublin, Ireland and Gothenburg, Sweden. Both have graciously offered to cover the direct travel and lodging costs for Lorrie and me. Lorrie would never forgive me if I went without her.

JASON: Of course.

Steve: So I'm very much looking forward, yeah, very much looking forward to that two months from now. I will be at the OWASP meeting in Dublin, Ireland on September 24th, and then two days later at the OWASP meeting in Gothenburg, Sweden on September 26th. They, of course, have Security Now! listeners. And if they have room, I'd love to meet more of our listeners at those meetings. So anyone who's interested should contact the organizers of those OWASP chapters and inquire about attending. It would be fun to meet more people and shake hands and so forth.

JASON: That sounds like a blast.

Steve: Lots happening, yeah.

JASON: Have some fun. That's great.

Steve: Very happy with the way things are developing and rolling out.

JASON: Absolutely.

Steve: So Sophos titled their research "RDP Exposed: The Wolves Already at Your Door." And for anyone who is interested, you haven't heard this before, Jason, but I announced GRC's link shortener a week or two ago, and I just talked about it in fact in the context of http://, and I saw that you brought up the nonsecure page. I used it to shorten the very long Sophos link to the PDF of their report. So for anyone who's interested, it's got just this podcast's number. So, well, actually sn724. So grc.sc/sn724. That will bounce you to the Sophos PDF titled "RDP Exposed: The Wolves Already at Your Door." And it's much longer. I'm summarizing it.

So they wrote in their summary: "For the last two months, the infosec world has been waiting to see if and when criminals will successfully exploit CVE-2019-0708." And of course we know that well. That's the BlueKeep vulnerability. They say: "The remote, wormable vulnerability in Microsoft's RDP (Remote Desktop Protocol), better known as BlueKeep." They write: "The expectation is that sooner or later a BlueKeep exploit will be used to power some self-replicating malware that spreads around the world and through the networks it penetrates in a flash using vulnerable RDP servers."

Now, we've talked about this a lot. I'll break from reading their summary for a second just to note that we haven't seen a worm. We've seen lots of exploits and proof of concepts. It's now been a couple months. And I've always thought that where it made sense for a worm to be created, like with the Exim email server vulnerability, and sure enough we got a worm pretty quickly because it took a week of connection to one of those servers in order to exploit that vulnerability, whereas with BlueKeep you can get right in with no authentication. And since Shodan will find them all for anybody, it's just a matter of, I mean, it's like easy pickings. You just choose one, you connect to it, and you set up cryptomining, cryptocurrency mining malware, and maybe close the door behind you so nobody else can kick your miner out. And now you start at mining Monero. So it just didn't seem wormable. Maybe mass exploitable, but not via a worm. And so far we haven't seen that. But anyway, RDP is the issue.

They write: "In other words, everyone is expecting something spectacular in the worst possible way. But while companies race to ensure they're patched" - and of course we know how that's gone, not - "criminals around the world are already abusing RDP successfully every day in a different, no less devastating, but much less spectacular way. Many" - now, here's the key. "Many of the millions of RDP servers connected to the Internet are protected by no more than a username and password, and many of those

passwords are bad enough to be guessed, with a little - a sometimes very little - persistence. Correctly guess a password on one of those millions of computers and you're into somebody's network. It isn't a new technique, and it sounds almost too simple to work, yet it's popular enough to support criminal markets selling both stolen RDP credentials and compromised computers. The technique is so successful that the criminals crippling city administrations, hospitals, utilities, and enterprises with targeted ransomware attacks and demanding five- or six-figure ransoms seem to like nothing more."

They have a chart which I copied into the show notes here showing the SamSam ransomware from 2015, no longer active, but its infection vector was RDP. Dharma, which we'll remember talking about in 2016, still active, using RDP as its infection vector. The Matrix ransomware, also 2016, still active, using RDP. BitPaymer, 2017, still active, using RDP. And we've been talking about it a lot, Ryuk, which has recently been the ransomware that infected all of the various municipalities, a bunch of them in Florida and the Georgia court system, using RDP. So we've wondered how they're getting in. Is it phishing? We know in one instance that it was tracked down to a phishing link. But in some cases the IT people are unable to say how. Well, if those networks had RDP exposed, and they had not patched - so remember, either dumb username and password, easy to get in, or you did not patch it after Microsoft said you must, then that's apparently one of the ways this ransomware is getting into people's networks.

So they continue: "All of which might make you think there must be a lot of RDP password guessing going on." Now, that would apply to patched RDP servers, remember. No need to guess with any of those millions that have not yet been patched. However, many more are patched.

They said: "Noting the popularity of RDP password guessing in targeted ransomware attacks, Sophos's Matt Boddy and Ben Jones and I set out to quickly measure" - "I" meaning the author of this - "set out to quickly measure how quickly an RDP-enabled computer would be discovered, and how many password guessing attacks it would have to deal with every day. We set up 10 geographically dispersed RDP honeypots and sat back to observe. One month and over four million password guesses later we switched off the honeypots, just as CVE-2019-0708 was announced."

So that was four million guesses against 10 honeypot RDP servers in one month before the announcement that there was a problem with them. On the other hand, as I said, once you know there's a problem, and once you know what the problem is, you don't need to guess anymore.

They said: "The low interaction honeypots were Windows machines in a default configuration, hosted on Amazon's AWS cloud infrastructure. They were set up to log login attempts while ensuring attackers could never get in, giving us an unhindered view of how many attackers came knocking, and for how long, and how their tactics evolved over the 30-day research period." The first honeypot to be discovered was found one minute and 24 seconds after it went live.

JASON: Wow. Wasting no time.

Steve: Yikes. One minute and 24 seconds after it went live. The last was found in just a little over 15 hours. So they have a chart here, and I copied it into the show notes, the time to first login attempt by honeypot. And so it happened that the one that was found took a minute and 24 seconds was in Paris. It took seven minutes and five seconds in Sao Paolo; 21 minutes and 29 seconds in Frankfurt; a little over half an hour, 35 minutes and 25 seconds, in Ohio. California took two hours and 38 minutes to be found. And then the chart goes up. Singapore was the last one, about 15.5 hours for the first login

attempt at a honeypot in Singapore. So again, that's unscientific. It doesn't represent a big sample size, but still it's interesting.

Then between them the honeypots received 4.3 million login attempts at a rate steadily increasing through the 30-day research period as new attackers joined the melee. So this is login attempts per day. We have a chart. The vertical scale is zero to, looks like it comes in at about 210,000 attacks per day. And it is clearly on the rise. So it looks like, once discovered, then the IP address of the server begins to spread or is being discovered by an increasing number of attackers. So there's a feeding frenzy out there on RDP servers, and this is what their data shows, to the point where they're clearly getting about 205,000 login attempts per day at the peak.

They said: "While the majority of attacks were quick and simple attempts to dig out an administrator password with a very short password list, some attackers employed more sophisticated tactics. The researchers classified three different password guessing techniques used by some of the more persistent attackers." And you could read more about them - they called them the Ram, the Swarm, and the Hedgehog - in the whitepaper.

So, they conclude, what to do? They say: "RDP password guessing should not be a problem." And of course I agree because you should never, as I have said, have an RDP server exposed. But they said: "It isn't new, and it isn't particularly sophisticated, and yet it underpins an entire criminal ecosystem." Once again, RDP password guessing underpins an entire criminal ecosystem.

They said: "In theory, all it takes to solve the RDP problem is for all users to avoid really bad passwords. But the evidence is they won't, and it isn't reasonable to expect they will. The number of RDP servers vulnerable to brute force attacks isn't going to be reduced by a sudden and dramatic improvement in users' password choices, so it's up to sysadmins to fix the problem."

And I suppose some RDP servers are not for sysadmin remote access. They're for random enterprise users to log in when you're on the road. And so they've probably deliberately dumbed-down the password, mistakenly thinking that it's okay to do so in order for their less security-conscious, more casual users to be able to log in. That's the wrong strategy.

They finish: "While there are a number of things that administrators can do to harden RDP servers, most notably two-factor authentication, the best protection against the dual threat of password guessing and vulnerabilities like BlueKeep is simply to take RDP off the Internet. Switch off RDP where it isn't absolutely necessary, or make it accessible only via a VPN, a Virtual Private Network, if it is."

And of course that's what I've been saying now from the beginning is, I mean, it is so simple to put your RDP server behind a VPN. It will not be visible. It will not be seen. VPNs can be secured in a way that doesn't inconvenience users, where you use certificates, and the VPN client in your user's laptop has the certificate that the RDP server must see and vice versa. That then establishes a connection to the RDP server, at which point the user logs in.

It's just, you know, RDP has been a troubled protocol. BlueKeep is just the latest example of a catastrophe with it. Anybody who understood that it was never safe to have RDP publicly exposed had nothing to worry about from BlueKeep. They could take their time patching it because it didn't matter. Really, just it's so simple these days to bring up an OpenVPN server. I mean, our routers all have them built into them now. So really give that some thought. And that's our show.

JASON: That's awesome. I just want to remind viewers and listeners, where can they go to read up on that?

Steve: Ah, right, grc.sc/sn724.

JASON: That's going to be in the show notes, and also have a direct link to the whitepaper that you were talking about. They have all that data listed inside that. That is really interesting. Doesn't take very long, does it. Like a swarm of bees.

Steve: Oh, my goodness. A minute and a half, wow.

JASON: Yeah, that is impressive. This is awesome, Steve. I want to, real quick before we end the show, call out that we have some guests who have joined us in the studio to watch the show live. Shelly and Cody are sitting right there on the left, and then we have Andrew...

Steve: Hey, guys.

JASON: ...on the right. And they've been sitting and enjoying the show throughout. So it's always fun when we have people visiting in the studio, and it's really great to have you guys here. Thank you for coming in. Anyone can, by the way. Tickets@twit.tv, if you email there, we can get you set up and then put out a chair for you in the studio while we record Security Now! and any other show that we do on the TWiT network.

But as for Steve, go to GRC.com. Everything you need to know that Steve's involved with, is doing, busy with every single day can be found there, GRC.com. SpinRite, of course, the best hard drive recovery and maintenance tool. You can get your copy there. Information, you talked about SQRL today, and that's super exciting. It feels like there's momentum right there. You can find more information about that at GRC.com.

Audio of the show, of course, can be found on there also. Transcripts of the show can be found there, as well. And again, that's GRC.com. If you want to go to our website, it's TWiT.tv/sn. There you can find all the links to our audio and video podcasts. The times of the show, which we actually record Security Now! live every Tuesday starting at 1:30 p.m. Pacific, 4:30 p.m. Eastern. I know it's probably out of date, but 20:30 UTC, does that sound about right, right now?

Steve: Sounds familiar, yeah.

JASON: Okay, all right. I forgot to check it before the show to be sure that there hasn't been some sort of a time switch from the last time I wrote that down. But anyway, somewhere around there. I guess do a little homework, and you won't miss the show. Steve, thanks a lot for allowing me to tag along. I really appreciate it.

Steve: Oh, allowing you? It's been a pleasure, Jason. Thanks so much for helping out.

JASON: All right. We'll see you all next week on another episode of Security Now!. Bye, everybody.

Steve: Bye.