What's Up With WhatsApp?

Description: This week, Leo and I discuss a classic bug at GoDaddy which bypassed domain validation for 8,850 issued certificates. Could flashing a peace sign compromise your biometric data? It's not only new IoT devices that may tattle - many autos have been able to for the past 15 years. McDonald's gets caught in a web security bypass; more famous hackers have been hacked; Google uses AI to increase image resolution; more on the value or danger of password tricks; and, finally, does the WhatsApp messenger incorporate a deliberate crypto backdoor?

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-595.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-595-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We're going to talk about WhatsApp. The Guardian says it's insecure. Moxie Marlinspike says no, no, it's secure, it's an implementation issue. Anyway, you know Steve will get to the bottom of that, plus all the security news. It's all next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 595, recorded Tuesday, January 17th, 2017: What's Up With WhatsApp?

It's time for Security Now!, the show where we cover your security, your privacy, your safety online and in the world at large with Mr. Spock himself, James Tiberius Gibson. He's got the thumb, he's got the "Y" thing. I just saw - what did I see? I just saw a video where a guy did that. I think it was a wedding video.

Steve Gibson: I did, too. And I was thinking, okay, thumbs correct.

Leo: We are now to the point where we are passing judgment on people's Vulcan hand gestures.

Steve: That's right. It only took us 12 years to get to this point, but that's [audio anomaly] something.

Leo: This is where nerds gather. I just want to, before you get going, it just broke, the story broke three minutes ago. President Obama has commuted Chelsea Manning's sentence - not a pardon, but a commutation. Chelsea Manning, of course, was she in the army? I think she was. I actually have a link that I can...

Steve: Didn't she have some intelligence role?

Leo: She was an intelligence agent and leaked to WikiLeaks a considerable amount of material, including, and I think the most important one, and the one that she really provided a public service, the video of a drone strike that showed really kind of an irresponsible use of a drone strike.

Steve: Wasn't that also - I have a hard time saying "she," but I guess we're supposed to.

Leo: She's a she. He was a he when that happened, and he's transgender now.

Steve: But wasn't that also the Abu Ghraib revelation?

Leo: It was, yeah, I believe so, yeah. So certainly - but, now, what's interesting is, of course, now you're going to wonder, Obama's got three more days in office, if he will also pardon - now, he would have to pardon Snowden. Snowden has never been on trial or sentenced. And I think that might be one of the big differences. Manning has served time on this.

Steve: So commutation means time served? It means you're now free?

Leo: Almost. She won't be free till July. So effectively she's shortened - her sentence has been shortened. Remember, she's committed or attempted suicide - she never committed it. But she attempted suicide several times in custody. And so I think some of this is humanitarian as much as anything else.

Steve: Boy, we really are in an interesting time of moral dilemmas, where, I mean, you know, like the question of what's right.

Leo: She was an army intelligence analyst, leaked classified material to WikiLeaks and was sentenced to 35 years. She would have been released in 2045. She attempted suicide twice and went on a hunger strike because she wanted gender reassignment surgery, which was granted to her. She was Bradley Manning, and she's now Chelsea Manning. She was imprisoned in 2010. So she's served six years, more than six years. 700,000 military files, diplomatic cables. You know, the same rhetoric was used about her as Snowden - traitor, endangering U.S. assets abroad and so forth. She pleaded guilty and apologized. And I think that may also have

something to do with the clemency. In any event, that's breaking news, just happened. Actually, I'm sorry, it's not July. BBC says May 17th. So what is that, four more months of her term, which is considerably less than her overall sentence. She was sentenced to 35 years in 2013.

Steve: Yeah.

Leo: Edward Snowden himself had tweeted that he asked, he said, "Mr. President, if you grant only one act of clemency as you exit the White House, please free Chelsea Manning. You alone can save her life." So there's the news, just came out moments ago. And now, on with the show.

Steve: So our title today is "What's Up With WhatsApp?" The Guardian broke the news late last week, I think it was Friday. And it just caused a huge storm of concern because WhatsApp, due to its connection to Facebook, is the most heavily used, supposedly private and secure messaging application on the planet; Facebook says 15 trillion messages handled per year. So it's a busy application. The problem is that there has been the allegation made that it contains a backdoor. So our main topic for the week will be taking a look at that. What does that mean? What's actually happening? And I know there's a great interest among our listeners because I got a flood of, oh, you know, what's this, is this right, incoming. So that's our title for this week. Otherwise we would have done maybe a Q&A. But I think this is an important thing for us to discuss.

So we've got a classic bug. Oh, my god, this one is just one for the history books, a classic bug at GoDaddy that bypassed domain validation for 8,550 certificates which were issued. Then the question of could flashing a peace sign compromise your biometric data. It's not only new IoT devices that may tattle on us. It turns out that many cars have been able to do that for the past 15 years, but it really wasn't ever on anyone's radar. McDonald's gets caught in a web security bypass the likes of which we have danced around and discussed even recently. And I heard you mention on MacBreak Weekly that the hackers have been hacked, namely Cellebrite. We'll talk about that. And one that I thought would really catch your interest, Leo, is that Google has come up with a technology to use AI to increase image resolution. That is without losing sharpness, actually putting information in that didn't exist in the original image, which you can't do. And a little bit more discussion, because I'm seeing a lot of questions about this still, on various spins of like, Diceware and using multiple English words and things.

So I thought, okay, we're to the point now where we've established enough foundation to lay down a formal definition of, I mean, sort of with almost mathematical formality, a formal definition of what it means to have a maximally secure password, and the test you can apply. And then we have a little bit of miscellany, not much; and then we're going to talk about the details of a design decision that WhatsApp made that has become controversial. So I think a lot of great things to talk about. And, oh, the picture this week, this was - someone named Steven Minnick tweeted this to me. This is a picture he took of the center console - you recognized it instantly.

Leo: Oh, yeah. Oh, yeah.

Steve: You knew what it was, because you have one, of this Tesla. I assume it was his

because it's from the driver's angle. The tweet says: "@SGgrc Steve, you look fantastic."

Leo: He's listening to Security Now!, "A Look Into PHP Malware." But he's got an album...

Steve: Last week's episode.

Leo: ...from somebody called Mashonda, who is wearing quite a low-cut negligee.

Steve: Ooh, goodness.

Leo: But I have to tell you, this has been going on for ages. Tesla does a terrible job of figuring out album art for podcasts. In fact, I have a picture I took just the other day. I'm listening to "The People's History of the United States" by Howard Zinn. And for some reason the album art it pulls up is Hamster Theatre with a guy with a Pinocchio-length nose.

Steve: I thought the podcast contained the album art.

Leo: It does. But the Tesla is not receiving it, I gather, from - remember, it's streaming it. It's not storing it. So it's streaming it from TuneIn. And so TuneIn doesn't send the album art, apparently.

Steve: Yeah. Anyway, if I looked like that, I would have a different relationship with myself than I do.

Leo: You wish you looked that good. You wish you looked that good, yeah. That's funny.

Steve: So, okay. As I said here in the show notes, the righteous bug of the New Year bites GoDaddy and allows the issuance of 8,850 certificates. Actually it turns out it ended up being a little more because of some interesting caching issues that they later found, without proper domain validation. Now, the first note here is that GoDaddy handled the entire mess perfectly and responsibly. Which doesn't make the bug any less wonderful. But unlike Woebegone, or WoSign that we talked about, that just completely tripped over themselves, this is also a textbook case of a responsible certificate authority discovering a problem, immediately responding to it, and then coming completely clean and doing all the remediation that you would want.

So Wayne Thayer is the senior Internet product and technology leader at GoDaddy. He posted in a Google Group, the mozilla.dev.security policy group, where this sort of stuff goes, on the 10th. So exactly one week ago today, during last Tuesday's podcast, this was happening, an incident report titled "Certificates issued without proper domain validation." So, first of all, that's like the worst thing a certificate authority ever wants to have to post somewhere. It's not something strange crawled out of the wall, it's "We

issued certificates that didn't get validated." And their whole job, I mean, that's the whole point of a certificate authority. So again, props.

So he writes: "On Friday, January 6th" - okay, so that was the previous Friday - "GoDaddy became aware of a bug affecting our domain validation processing system. The bug that caused the issue was fixed late Friday, the same day." So instant response in terms of, like, knowing about it and immediately fixing it.

"At 10:00 p.m. PST on Monday," he writes, so three days later, January 9th, "we completed our review" - which they were probably doing all weekend - "to determine the scope of the problem and identified 8,850 certificates that were issued without proper domain validation as a result of the bug. The impacted certificates will be revoked by 10:00 p.m. PST on Tuesday" - that is, 24 hours later - "Tuesday, January 10th, and will also be logged to the Google Pilot CT log."

That's Google's certificate transparency system, which is on my list of things to get to for the podcast because it's a terrific idea. Basically it's a publicly postable and viewable log that the concept would be that all certificate authorities would publish the certificates they issue as they do, and other transactions about it, the idea being that then there is a - it's called "certificate transparency" - a single place where everything can be found to both create visibility and accountability and allow much better, you know, basically to sort of close an aspect of this open and supposedly public process, but which still isn't getting enough light shined in it. So that'll be a topic that we'll be getting to.

So under "detailed description," Wayne continues. He says the week before that, so on Tuesday, January 3rd, "one of our resellers, Microsoft, sent an email to a GoDaddy notification account" - which I guess from the context was the wrong place to send it - "and two GoDaddy employees. Due to holiday vacations and the fact that the issue was not properly reported per our guidelines, we did not become aware of the issue until one of the employees opened the email on Friday, January 6th, and promptly alerted management. The issue was originally reported to Microsoft by one of their own customers and was described as only affecting certificate requests" - get this, this is strange - "when the DNS A record of the domain was set to 127.0.0.1."

Which is, okay, now, I guess you could do that by mistake. So the reason you would do that is if you wanted to get a certificate for some local internal purpose. But that's not being issued anymore. That is, one of the changes that we've seen in 2017, and we've talked about this before, is that certificate authorities are going to stop issuing certificates for private domains, only doing it for public domains. So that's clearly not what GoDaddy intends. Because I should just mention that 127.0.0.1 is of course the localhost IP, meaning it's an IP that always resolves to the system that's asking the question. So it would be, well, for example, I have one that I issued myself for my own purposes. It's www.steve. And that refers to...

Leo: That's a self-signed certificate; right?

Steve: Precisely.

Leo: So anybody could do that. There's no...

Steve: Yes, yes. And so what's different here is that they're getting one - and so, for

example, I've arranged for my browsers and other clients to trust that certificate so that I'm able to do things locally to test the security that I then deploy publicly at GRC. That allows me to operate, to have a staging area. Which is probably what this person was wanting.

Leo: It's a fairly routine thing.

Steve: Right, right.

Leo: So you wouldn't normally go to GoDaddy to do it. That's what's weird.

Steve: Correct.

Leo: You just do it yourself.

Steve: Correct. And now the CAB guidelines say no more issuance of certificates for nonpublic IPs. So they continue: "An investigation was initiated immediately, and within a few hours we determined that the problem was broader in scope." Oh, boy. "The root cause of the problem was fixed via a code change at approximately 10:00 p.m. on Friday, January 6th." So same day they found out about it, they plowed into it, found the problem, and immediately fixed it.

On the next day, on Saturday - so yes indeed, they were working the weekend - January 7th, "we determined that the bug was first introduced on July 29th, 2016, as part of a routine code change intended to improve our certificate issuance process. The bug is related to our use of practical demonstration of control to validate authority to receive a certificate for a given fully-qualified domain name. In the problematic case, we provide a" - and so here's part of the problem. "We provide a random code to a customer and ask them to place it in a specific location on their website. Our system automatically checks for the presence of that code via an HTTP and/or HTTPS request to the website."

And at first I was thinking, wow, how could HTTP be secure? But on the other hand, I'm sure they're taking that into account. "If the code" - that is, this random nonce, essentially - "is found, the domain control check is successfully completed. Prior to the introduction of the bug, the library used to query the website and check for the code was configured to return a failure if the HTTP status code was not 200." We've talked about this before. When the browser makes a query to a remote server and says blahblahblah.html, in the response headers there is an HTTP "200 OK" is the common phrase. And there are different things that can be returned. For example, it might say "301 redirect," in which case another header will tell the browser that this asset has moved to a different URL, and the browser should then ask again at this new location.

So those little return codes are very handy. And of course the famous one is the 404, which people typically encounter. We used to more in the past. Now, of course, when a domain is abandoned, you generally get a squatter who sticks some junk there, just so you get something. But if you just put, like, if you put in a completely erroneous long URL yourself, then you can get a 404 because the page you've asked for is not available. So that 404 code comes back, that HTTP 404, and then it typically says "not found" afterwards, meaning we looked for that page, but couldn't find it.

"A configuration change to the library caused it to return results even when the HTTP status code was not 200." Now, here it is. "Since many web servers are configured to include the URL of the request in the body of a 404 'not found' response, and the URL in question also contained the random code, any web server configured this way caused GoDaddy's domain control verification to complete successfully." This is just such a sublime bug.

So what that meant was that, if the domain being checked wasn't owned by the person who asked for a certificate - excuse me, I've got a little scratch in my throat. If the domain for which a certificate was being requested was not owned by the requestor, GoDaddy would send back to the requestor saying, "Here is a special code. Put this on the following page on your site and let us know when you have." So whether the person did or not, they would say okay. And then GoDaddy would ask for that page containing that code, but the request also contained the code. So when the site that wasn't owned by the user said, huh, I don't have any page like that, if in trying to be helpful that web server's 404 "not found" said, oh, you know, sorry, we could not find this URL you just asked for, and displayed that in the error page, the modified library that GoDaddy was using since July would say, oh, and find the code in the error page rather than in a legitimate page, and pass the domain validation test. Whoops. Wonderful bug.

So Wayne continues: "We are currently unaware of any malicious exploitation of this bug to procure a certificate for a domain that was not authorized." So in other words, they don't know that it was abused. So it wasn't found...

Leo: Probably wasn't. I mean - right?

Steve: Right.

Leo: Because nobody knew it was there.

Steve: Exactly. Exactly. "The customer who discovered the bug revoked the certificate they obtained" - oh, so maybe there was one test. And, he said, "and subsequent certificates issued as the result of requests used for testing by Microsoft and GoDaddy have been revoked. Further, any certificate requests made for domains we flag as high risk" - for example, Google.com - "were also subjected to manual review." So if during that time any high-value, high-risk domains happened to be reissued, those they immediately, you know, they read through the whole list and immediately were responsible.

"We have re-verified domain control on every certificate issued using this method of validation in the period from when the bug was introduced until it was fixed. A list of 8,850 potentially unverified certificates" - it says "potentially unverified" - "representing less than 2% of the total issued during the period" - because I should mention that most certificate authorities, and we've discussed the various means of validation in the past, they'll give you multiple ways of doing it. It's like take a file, stick it on the root, do this, do that. Make this appear in a text record in your DNS records. So they typically give you a range of things to do.

And you'll remember that, back when I was playing with trying to get a third party to scan GRC.com when I had received a report of there being some sort of a cross-site

scripting vulnerability, I had trouble doing that because - I don't remember now why. Oh, it's because I don't use a default home page. I always redirect anyone coming in to just the root, you know, backslash, I always redirect them - or rather, forward slash. I always redirect them to intro.htm, which I've long used. And their systems wouldn't follow a redirect. Which maybe was good for security, but it wasn't something I could easily fix. So anyway, so as a consequence, there were other ways I was able to do it.

So then they conclude, saying: "Additional code changes were deployed on Monday and Tuesday to prevent re-issuance of certificates using cached and potentially unverified domain validation information. However, prior to identifying and shutting down this path, an additional 101 certificates were reissued using such cached and potentially unverified domain validation information, resulting in an overall total of 8,951 certificates that were issued without proper domain validation as a result of this bug."

So as we know, anyone can make a mistake. GoDaddy's response is textbook perfect. And finally, under "Next Steps," he concludes, saying: "While we are confident that we have completely resolved the problem, we are watching our system closely to ensure that no more certificates are issued without proper domain validation, and we will take immediate action and report any further issues if found. A full post-mortem review of this incident will occur, and steps will be taken to prevent a recurrence, including the addition of automated tests designed to detect this type of scenario in the future. If more information about the cause or impact of this incident becomes available, we will publish updates to this report. Wayne Thayer, GoDaddy."

And again, it doesn't get any better than that. I mean, you know, talk about a different response from WoSign, who said, "Oh, okay, yeah, we revoked that one." And it's like, "Okay, but this was a big problem you had. What else was affected?" "Oh, well, I guess we could look." It's like, no. Don't use them. And for what it's worth, I still love Hover. They are my goto certificate authority. Or, I mean, not certificate authority, domain host. And of course DigiCert is my certificate authority of choice. Was I saying CA? I was saying CA this whole time, wasn't.

Leo: I hope so. Oh, instead of domain registrar.

Steve: Yeah.

Leo: Domain registrar. Please s/ca/domain registrar/g. All fixed. Oh, wait a minute, this isn't text. Just a little grep. Yes.

Steve: So anyway, just beautiful, beautiful work for handling domain registration. And again, I just want to - I do want to give Hover props. In fact, I just allowed three domains to expire on GoDaddy. I was trying them for a while, and they're just not my cup of tea. Also they're sort of hypercommercial, and I really just want somebody that does, you know, that's got good coverage, good policies, and a long track record. And of course Hover is based on Tucows. It's a service of Tucows. And so I'm happy there. But again, couldn't ask for better performance from those guys.

I don't know what she's doing with her tongue here, Leo. It's rather...

Leo: Okay. That's a bad way to start.

Steve: It's rather distracting.

Leo: She has a very hyperactive tongue.

Steve: Should I know her?

Leo: Oh, yeah, it's Miley Cyrus. Don't you know Miley Cyrus?

Steve: No, Leo.

Leo: Oh, come on, Steve.

Steve: I don't know her or her tongue.

Leo: She's very famous for, well, you know Billy Ray Cyrus, her dad.

Steve: Okay, first of all, I know the name. I absolutely know the name.

Leo: Yeah, she's a pop star.

Steve: But I couldn't pick her out of a crowd.

Leo: Yeah. She's also on "The Voice" right now. She's one of the judges. And she has really a - there must be a name, I would love to know what it is, for an agile and lengthy tongue.

Steve: It's freaky.

Leo: Yeah.

Steve: She's rather inked up, also.

Leo: Yeah. Well, she's under 30.

Steve: Has she got, like, crib notes on her left hand? What does that say there?

Leo: Those probably are. Yeah, that's probably something she wrote, and then - it's hard to distinguish those from the tats. But the point of this photo is she's flashing the peace sign.

Steve: Yes. And what I have noticed is, well, we've all noticed camera resolutions are skyrocketing. And I am sometimes still taken aback when I zoom well into a photo and the image remains super sharp. Because, I mean, in the old-school days you're used to zooming in, and it gets all blurry. Except that the resolutions are so crazy now that we are picking up a lot of additional information. So the question is raised, could flashing a peace sign - or a Vulcan hand grip. That's now a problem.

Leo: I can see your fingerprint right now. Right now.

Steve: Could that lead to biometric theft? And I would have to argue, with sufficient resolution, certainly if somebody had a good lens and image stabilization, and the lighting was correct, and meaning that you were able to pick up the 3D nature of the ridges, it's probably possible. We have seen an instance where a distant photo of an old-school mechanical lock key was taken with a state-of-the-art camera. It was modeled. It was rotated to remove all the perspective distortion. And then a key was cut from that, and it worked the lock.

Leo: It's how the TSA master key was cracked. Pictures.

Steve: Yup.

Leo: Now, if somebody had your fingerprint, though, I mean, most fingerprint readers you can't just put the - I think you'd need more than just an image of the fingerprint, wouldn't you? Maybe not.

Steve: Well, yeah. And we've talked about that a little bit. And so the hope was that, by using - technically the term is "projective capacitance." That is, when you put your finger down, for example, on the Apple capacitive touch sensor, we heard that it wasn't reading just that surface layer, that there was some penetration. And so it was reading the meat a little more than just the ridges that are ruffled. And but on the other hand, we also have seen successful spoofs using gummy technology, using like conductive rubber that has ridges on it in the pattern of a fingerprint. So I don't think we can rely enough, unfortunately, on the technology, for example, verifying there's a pulse and maybe checking your pulse ox to say, okay, yes, this person is alive, instead of just being a gummy bear that's had a fingerprint etched on top.

Leo: Yeah, I think there's infrared. They read the heat and, I mean, which is not to say you couldn't create a prosthetic that would work. But Steve, this is a part of a much larger picture, which is essentially that basically your soul can be stolen now. And if somebody has an image of you, they can make it do anything they want. I mean, you can no longer trust photographs or video. You can, I mean, it's...

Steve: Well, we saw a young Princess Leia.

Leo: Right, right.

Steve: And, you know, she never said what she said.

Leo: Right, right. Commander, what is it, Commander Tarkin was reproduced in the newest one? I haven't see it yet. But I hear that's - so, yeah, exactly. We're in that era.

Steve: Yeah, yeah.

Leo: And it's people like you and me, who have thousands of hours of video widely accessible, I think, yeah, I think your fingerprints are the least of it. You could be recreated.

Steve: Well, and that takes us into this next article perfectly because Forbes got a lot of attention this week from their Thomas-Fox Brewster, the guy who covers crime, privacy, and security in digital and physical forms for Forbes. And he did some research that revealed something we may have sort of been sort of dimly aware of, but not focused on. And that is that, while our IoT devices that we've just been talking about are all in the news, for example, we just recently were talking about the Amazon Echo and how it has been subject to a subpoena, and in fact it was that that brought this to Thomas's awareness.

He writes: "The rapid spread of connected devices that can listen and locate has been a boon for law enforcement. Any new technology hooked up to the web has the potential to become a surveillance device, even if its original purpose was benign, as shown in a 2016 Arkansas murder investigation where Amazon was asked to hand over audio from a suspect's Echo." And so he broadens this to note that "such information and much more, I've learned," he writes, "has long been retrievable from cars."

Now, of course, this is also fodder for investigative movies and all kinds of things, where you have GPS in a car that is building a log of where you go. But he did some court document research and revealed a 15-year history of what's been dubbed "cartapping," where almost real-time audio and location data can be retrieved when law enforcement order vehicle tech providers to hand it over. And he notes that, for example: "One of the more recent examples can be found in a 2014 warrant that allowed New York police to trace a vehicle by demanding satellite radio and telematics provider SiriusXM" - and first when I saw it I thought, wait a minute. SiriusXM is one way. It's downward traveling satellite. You're not sending anything up to the Sirius satellite.

But, "The warrant, originally filed in 2014, but only recently unsealed" - and in the article he publishes it in full - "asked SiriusXM to activate and monitor as a tracking device the SiriusXM satellite radio installed on the target vehicle for a period of 10 days." The target was a Toyota 4Runner wrapped up in an illegal gambling enterprise. And as Thomas dug into this, he found that, well, he talked to Sirius, who told Forbes it "complied with the order and did so by switching on the stolen vehicle recovery feature of its connected vehicle services technology, which is only available in a subset of cars it supplies." And

he notes that the satellite radios alone cannot be tracked as the telematics services can. And so on. In fact, his article goes to great length to enumerate many of these sorts of things.

So it's just sort of, again, another heads-up that, when law enforcement discovers a means for acquiring data, we can no longer think, oh, maybe they're not going to pursue it, or they're not going to try to get it. What we're seeing is instances - and in fact this brings us later to the WhatsApp question. What we're seeing is that anything available, someone's gas meter, which is an IoT device, or of course the Amazon Echo, or law enforcement will examine the technology in the cars believed to be used by suspects, obtain a court order, and use whatever technology is available there in order to pull data from it. So Leo, to your point, this is the world we are in today.

Leo: Yeah.

Steve: And I'm doing nothing very interesting in my car, so I'm not that concerned. Of course my car has the advantage of being 15 or 16 years old, and so it's completely disconnected. But I'm sure...

Leo: Yeah, I'm the exact opposite, of course.

Steve: Correct. Well, and because of its age and mine, I know I'm going to outlive mine. And so at some point I will be upgrading, and I no doubt will be, you know, I mean, I see these things when I travel somewhere and rent a car, or am a passenger in somebody else's, all this stuff happening. And so I'll have that eventually. I guess the only takeaway would be, if somebody is really concerned about privacy, modern car technology creates a problem because, well, especially now we're going to have cars, as you have mentioned, as we talked about on this podcast and I heard you talking about elsewhere, the National Highway Traffic Safety Board will eventually be moving inter-vehicle communication from optional to mandatory. So we're going to be driving connected computers around. And so another aspect of privacy disappears. And of course we've talked about license plate scanners that are increasingly present, so that this information is just - it's out there.

Leo: I feel like we're all living in a Philip K. Dick novel at this point. I mean, really, science fiction is coming true.

Steve: I know. That's exactly right.

Leo: Yeah.

Steve: Yeah, it really is. I mean, some of the things we used to talk about on this podcast, it would have been science fiction 10 years ago, while we were doing the podcast. Now it's like, oh, yeah. Yeah, the guy's power meter finked on him.

Leo: Moving fast, it's moving fast.

Steve: Because he apparently washed down the patio, and we know how much water he used at 2:00 a.m.

Leo: Yeah.

Steve: What were you doing in the middle of the night?

Leo: I had to fill the hot tub, man.

Steve: So McDonald's website was found to have a problem which they didn't acknowledge. And after the person gave them plenty of time, I think six months or more, to respond, when he couldn't get a hold of anybody who cared, he said, okay. And so he laid it all out. And our listeners will quickly understand the problem. His post was titled, the blog post, "Stealing Passwords from McDonald's Users." And essentially he worked out all the details required to do that, where he can go to anyone's computer who has ever logged onto McDonald's and display their password in cleartext. This is a consequence of a series of mistakes. In order for that to be true, you have to really mess things up in terms of your web services.

But there's one place where this starts. And the title was "Reflected XSS" - which of course is cross-site scripting - "through Angular JS" - which is a well-known JavaScript Library - "sandbox bypass causes password exposure of McDonald's users." What he first saw was - and this is a mistake kind of related to what happened with GoDaddy, but this is similar to what caught me. Because remember that I, when I was asking people to enter their transaction code in order to look up a previous purchase of SpinRite, if they mis-entered it, I would return an error and say, oh, sorry, we couldn't find that transaction code. And to be helpful I was returning it filled into the form so they could look at it and check again and just fix the typo. Well, just that, that helpfulness, created a potential vulnerability because what that allowed was a user to control what the web browser displayed.

And just last week we were talking about the dangerous power of PHP, where if an attacker could somehow arrange to cause a victim or target web server to display a page containing PHP code that the attacker provided, then in the display of the page, as it went from the web server and was intercepted by the PHP parsing module, if it encountered PHP, it would execute it. And if it was the attacker's code, that's not good.

Well, the closely related problem is the same kind of thing at the client side, whereas if you don't have PHP, pretty much everybody is running JavaScript. I mean, even I gave up blocking JavaScript some time ago, as we all know. I dropped using NoScript because it was just becoming too much of a pain and switched to uBlock Origin. So similarly, if your browser gets a web page, it scans it for any JavaScript embedded in the page's content that it should execute. So if the attacker can arrange to put their own script tags in, then you are in trouble.

So what I had not done in my case was properly "escape," as it's called, escape the special characters, like the less-than sign and the greater-than sign, which are used to

bracket the keyword "script," because, for example, if I had simply - and I do now - convert the less-than sign to an < that's the HTML code for less-than, and similarly the greater-than sign to an >, then it'll still display as if it were the brackets, but it doesn't parse. They aren't actually brackets. And so JavaScript never gets invoked.

Anyway, those things are missing from McDonald's corporation website in the search results. When you search the McDonald's site, and whether or not it finds anything, it comes back saying "X number of search results were found for," and then, yes, it feeds back on the page what you gave it to search for. Then it even fills it into a form field to be helpful, and then asks do you want to search, and then it gives it to you again in the news section. So that's the beginning of a detailed dive that I won't go through in infinite detail here because we've pretty much covered the high points.

But Angular JS is supposed to be sandboxed. And it turns out the sandboxing isn't done right, and it can be bypassed to break out of a sandbox. And remember the extra danger of code running that appears to be from the site. Remember there is the same-origin policy, which is strictly and rigorously enforced by our web browsers. And that is that code from a domain, from a so-called "origin," is only able to communicate with its own origin, that is, where it came from. That's what prevents, for example, code in ads from being able to attack the sites that serve the ads. The code in ads that came from an ad server can only make queries back to that same ad server. So there's a strict compartmentalization.

But in this case, where you put something into a search term, and it's displayed by the server back to you, that code came from the server. Which means that the attacker, the site is not protected by the same-origin policy because it is from the same origin. It came from that server via the form being responded to. And so anybody who wants to who's clever can sit there poking at it, doing tests, and work their way through the security. And there are many things wrong. For example, they store - oh, I can't even believe I'm going to say this. They store the user's password encrypted in a cookie on the user's browser. It's like, why? Why would you do that?

I mean, so we're looking also at an amateur who implemented the logon process and authentication process. You would never encrypt the user's cookie, I mean, encrypt the user's password and store it in their cookie. It's like, no. Accept the password, hash it the same way - well, there's no hashing going on, obviously, this is so screwed up - and then check it against the previous hash from the time that the user created the account. We all know how this is to be done properly. None of that happens here.

So there's crypto present. The initialization vector and the key are static across all users. It's not per user as it absolutely should be. I mean, it's just mistake after mistake. I was just reading this thing and going, oh, you know, this is not something you repair. You just flush this and get somebody who actually knows how to build a website. So unfortunately they hired somebody who was unable to implement security in a useful fashion, and then didn't listen to a responsible individual jumping and down and waving his arms, saying, "Hey, you've got a real problem here." And of course, if this is all that was found, we know there's, I mean, this is sort of probably the tip of the iceberg.

So you don't want a situation where somebody can drop a magic incantation into your browser and up pops the JavaScript alert box with your password in the clear, which it decrypted from your cookie, which for some reason that site stored in your browser. It's like, no. Where do you begin with what's wrong with this? So anyway, again, these guys are doing it wrong.

And this you talked about a little bit on MacBreak Weekly. Or I guess, no, I think Rene

said that you were aware of it, but you hadn't talked about it. And that is that, although I did see the amount of data, which was rather substantial, Leo, we've talked about Cellebrite before. They're the Israeli, well-known, very popular in their circle...

Leo: Among law enforcement.

Steve: Exactly, law enforcement.

Leo: Authoritarian regimes everywhere.

Steve: Exactly, provider of cell phone hacking technology. So Motherboard wrote: "Motherboard has obtained 900GB" - okay, so that's not some usernames and passwords, nearly a terabyte - "of data related to Cellebrite." Motherboard was directly contacted, by the way, by the attacker who extracted the data from Cellebrite. Motherboard continues: "...one of the most popular companies in the mobile phone hacking industry. The cache includes customer information, databases, and a vast amount of technical data regarding Cellebrite's products. The breach is the latest chapter in a growing trend," writes Motherboard, "of hackers taking matters into their own hands, stealing information from companies that specialize in surveillance or hacking technologies.

"Cellebrite is an Israeli company whose main product, a typically laptop-sized device called the" - get this - "Universal Forensic Extraction Device (UFED), can rip data from thousands of different models of mobile phones. That data can include SMS messages, email, call logs, and much more, as long as the UFED user is in physical possession of the phone."

And in fact there was a really sort of interesting photo that went along with this, that showed sort of a filing cabinet with just hundreds of phones shown, like in their repository. And I've seen such things in hard drive data recovery facilities, where they maintain a massive store of spares for drives that die because their main board fries or something. And they go, oh, yeah, we have one of those, and they swap the main board in, and the customer's back in business. This is like that. It's like, let's see. And of course the phone may be 10 years old that law enforcement wants to crack. And so it's like, oh, do we have one of those? Yep, it's over here down Aisle D. So, yeah, they've got quite a collection.

So they write: "Cellebrite is popular with U.S. federal and state law enforcement; and, according to the hacked data, possibly also with authoritarian regimes such as Russia, the United Arab Emirates, and Turkey. The data appears to have been taken, at least in part, from servers related to Cellebrite's website. The cache includes alleged usernames and passwords for logging into Cellebrite databases connected to the company's my.cellebrite domain. This section of the site is used by customers to, among other things, access new software versions.

"Motherboard verified the email addresses" - I thought this was kind of clever - "in the cache" - meaning this 900GB of data they received - "by attempting to create accounts on Cellebrite's customer login portal. In the majority of cases, this was not possible because the email address was already in use." Meaning there's already an account there with that email address. So that's sort of a nice negative feedback means of verifying that, yes, this data appears to be authentic.

"A customer included in the data confirmed some of their details." So Motherboard found someone they knew and said, "Is this you? Does this sound right?" It's like, ooh, yeah. So the [audio anomaly]. "The dump also contains what appears to be evidence files from seized mobile phones, and logs from Cellebrite devices." So the hackers got hacked.

And Leo, get a load of this one. Google calls this RAISR - as in sharp and edge - R-A-I-S-R, an acronym for Rapid and Accurate Image Super Resolution. This is a research paper. A couple interns at Google in research and a few others teamed up. The abstract of this paper reads: "Given an image, we wish to produce an image of larger size with significantly more pixels and higher image quality. This is generally known as the SISR, [S-I-S-R] problem." Stands for Single Image Super-Resolution, meaning you don't get multiple images where you could then infer from multiple images a single higher image. You get one low-resolution image, and you are being asked to bring it to super-resolution.

"The idea is" - and get this - "is that, with sufficient training data" - and this is where I was thinking, when you talked about we're pretty much into science fiction and some other world with AI now - "with sufficient training data, meaning corresponding pairs of low and high resolution images, we can learn" - in the training and AI sense of "learn" - "a set of filters (in other words, a mapping) that, when applied to a given image that is not part of the training set, will nevertheless produce a higher resolution version of it, where the learning is preferably low complexity.

"In our proposed approach," they write, "the run-time is more than one or two orders of magnitude faster than the best competing methods currently available, while producing results comparable or better than state of the art. A closely related topic is image sharpening and contrast enhancement, in other words, improving the visual quality of a blurry image by amplifying the underlying details.

"Our approach additionally includes an extremely efficient way to produce an image that is significantly sharper than the input blurry one" - so not only could it increase the resolution without getting blurry, but it could also just, like, using AI, essentially what they have done is they have trained something in what blurry pictures look like, and it's good enough to remove the blur, to learn what the original sharp picture was. And if it knows that, then it could fix a blurry one, or scale it up to whatever size you ask. So it's just - I'm thinking, okay, I really am beginning to feel my age when this sort of stuff is just happening.

And so finally they conclude, saying: "We illustrate how this effective sharpening algorithm, in addition to being of independent interest, can be used as a pre-processing step to induce the learning of more effective upscaling filters with built-in sharpening and contrast enhancement." So this is sort of the beginning of this next generation of image processing where one of the things that always bugs all of us who listen to this podcast is where a cheesy movie takes a satellite image of basically a furry blob in the desert, and then someone says, "Can you enhance that image?" "Oh, yes, sir." And then, you know, types a few buttons, and you see this wonderful thing scanning back and forth, and cubes are zipping around, and things are scaling in, and it goes [sound effect], and with each one of those [sound effect], it's like getting sharper and sharper, and magically emerges the license plate that you can read from a keyhole satellite at 50,000 feet, who knows.

Anyway, it's looking like that's not so farfetched because, when you think about it, different original images will have slightly different blurs. And so there are things you're going to miss. That is, if there was a defect in the image that the blur masked, well, the AI would have no way of knowing that, that is, I mean, if the information was really

removed. But there's been a lot of work on anti-blurring filters. And now the point is they're going beyond an algorithm to something that is actually a knowledge base which has been trained by looking at blurry and not blurry, or a different size, and learning how to go from one of lower quality to higher.

And this all, by the way, was couched in various articles that picked up the story, talking about how Google might be, and this is a long way away, of course, but it was sold as a means of reducing the communications bandwidth. That is, you send thumbnails, and this magic brain running in your browser, you know, Chrome Triple-Plus, just expands them to the proper size, and it looks just fine. And even if it is Myrus, who, Smiley? Sniley? What is her name? I forgot.

Leo: Sniley Myrus. No, no. Miley Cyrus.

Steve: Miley Cyrus, yes.

Leo: She was TV's Hannah Montana. Come on, Steve, you must have watched that.

Steve: Eh, no. No, I was busy with "Stargate SG-1" when she was dancing around or whatever she was doing. So I got a question, I just picked this one out of the bag of many because our discussion that we've been having recently of passwords sort of brought this to the fore. Rodrigo Barrouin said: "You said that a random character password was better than one with random words. Is this true of the Diceware system, even if longer?"

So the Diceware system is one which is basically sort of an easy and fun-to-use system for choosing words at random in order to build a larger composite password. And many people ask this or related questions. And remember it was last week we were talking about how, whatever it was, I can't remember the example I used. Something or other 123.

Leo: Horse, stable, cat...

Steve: Yes, thank you, horsestaple123.

Leo: Something like that, yeah.

Steve: Yes. And the fact that I could say it that quickly was a clue that we had a problem here because horse and staple are objects which are known. And so their actual - the entropy they contain is dramatically less than the entropy that number of characters could hold, which is the problem. And then I pronounced a 12-character piece of gobbledy-gook that took me 15 seconds just to get it all said, which similarly demonstrated that it actually had more entropy. There was no way I could express that to our listeners any faster, which was, again, a clue.

So I thought, okay, let's formalize this. And here it is. If there exists ANY - and I put this

in caps in the show notes. If there exists ANY strategy for in any way short-circuiting a brute-force attack, then, while the system may still offer sufficient security, it does not offer maximum possible security.

So again, if there exists any strategy for in any way short-circuiting a full brute-force, basically try everything attack, then even though the result may still offer sufficient security, it does not offer maximum possible security. Even my own Password Haystacks method has been validly attacked by those who said that haystack patterns could be checked for. And those people are absolutely correct. Checking for haystack patterns would be a strategy that would be better than sheer, blind, brute force. So in terms of composing a truly maximally difficult to crack password of a given length, only if every character composing a string's password is equally probable to occur, randomly chosen, and without any interdependence among or between other characters, do we have both maximum entropy for the length and, by definition, no possible strategy for reducing the search space below what's required for a full brute-force attack.

So that's the formalization. Any strategy, if you can look at it and invent a way of making it quicker, you're in trouble. So clearly horsestaple123, well, dictionaries exist. And we know that people tend to like actual words. They may change the capitalization around, and that will slow down the attack. But so imagine a brute-force system that was really well designed. The last thing it would try was to assume a super-competent password chooser. It would hope that a human used horsestaple123. And so it would try those. It's got its dictionary. It would use the words in the dictionary and put them in different orders and intersperse different numbers. And then, if it's really up to speed, it would think, huh, I wonder if this guy might listen to Security Now! and did the Haystacks approach to lengthening the password.

And so, again, before trying, before dropping to just, okay, I give up, I'm going to try everything, it would do things like append dashes to the front and the back, a bunch of them, and then asterisks, and then stick them in the middle and, you know, do haystack-y things. The point is, if you look at it, and you can invent a way, and an algorithm could check for that, then that's not maximum entropy because a committed cracker will check for that. That's what we're learning. The cracking technology does do that increasingly. And so the only safe place to hide is with maximum entropy. Anything you do, even a bunch of words chosen at random, sorry, we've got a dictionary. Those are words. We're going to try those before we go off into the blue and just assume the worst case for the cracker, which is a maximum-entropy password. So that's of course what we want to use.

Leo: Moving on.

Steve: Yes. So just one little bit of miscellany. The question was raised, and we discussed what I had heard Reince Priebus say Sunday before last when he was criticizing the DNC's handling of email and the DNC hacks and so forth, claiming that John Podesta's email password was "password." Well, it turns out that we sort of know where that came from. The good news is it wasn't exactly "password," but it fails the high-entropy test by a long shot. It's p@ssw0rd.

If we assume the validity of this piece of information from WikiLeaks, this was Eryn Sepp, who I did track down. She is an administrator with the DNC. And this email that WikiLeaks has is from her to John Podesta, saying, with the subject "2 things," says: "Though CAP" - C-A-P, whatever that is - "is still having issues with my email and computer, yours is good to go." And then it says "jpodesta" on the next line, and this

"p@ssw0rd" with an "@" sign and a zero, which I think is her giving John Podesta his email account login. Now, presumably somebody who was security conscious would change their password.

Leo: That was his login for his computer, though, not his Gmail.

Steve: Oh, really?

Leo: Yeah. Read it closely. And it's, by the way, we do it, too. In fact, the default we use is "changeme." And then you're expected to take that and change it. So again, we do know what the password was. It wasn't particularly good. The Gmail password is apparently the combination of the word "runner" and four digits.

Steve: Ah, so it was not p@ssw0rd.

Leo: No.

Steve: Okay.

Leo: This is just - and by the way, so what if it was? That's blaming the victim. Doesn't mean he wasn't hacked by Russia. And anyway, do we know if it was a spearphishing attack? I mean, was it just somebody brute-forcing his account, or was it a spearphishing attack?

Steve: Yeah. And, see, this is the problem with any of this kind of coverage is that we just don't know.

Leo: We don't know.

Steve: Yeah. And so it's just not - it is, in fact, it's unknowable, unfortunately. I did have a...

Leo: I thought it was a phishing email. I was pretty sure.

Steve: I definitely heard that, Leo. But so that's what was said. And in fact, it might even have been somebody over on the DNC side who said that, not somebody who may have had an axe to bear, to grind, or do something with.

Leo: IT guys routinely will do that. Here's your password. And it's some loopy password like "changeme." That's what we use. And you should change it.

Steve: Right, right, yeah. And so that's just enough to get you to log in, in order to make the password change, yes.

Leo: Anyway. It says, if you read it closely, it says "Windows 8 password." He's telling him how to log into his Windows 8 system. "Your computer is good to go." And then he warns you, the Windows 8 system is very different from what we had back at the White House.

Steve: I did see all that, yes.

Leo: Yeah. So, you know, this whole story came from, well, I don't even want to go into it. And I don't think it's even germane. If the Russians hacked his email, whether they did it easily or with difficulty doesn't change the importance of a foreign government attempting to change the results of our election.

Steve: I was just following up because I had heard that, and there was some question about it. And then when I saw this, I thought, oh, maybe it was actually "password."

Leo: Now they're saying "runner," the chatroom's saying "runner" wasn't even for his Gmail. That was his iTunes, his Apple account. So I don't know.

Steve: It's so sad that we know all of this, though.

Leo: Well, because his email was leaked. But I have to say, if you read the email, besides being embarrassing, it wasn't particularly revelatory.

Steve: No, and I think in general that...

Leo: I never saw a smoking gun in there.

Steve: Right, right, right. So I got email from Tim D., who said he's near Detroit, Michigan. And this was following up to last week's discussion, and this is a tip that I had never really thought about. But when he said it, it's like, it makes so much sense. He said - so the subject was SN-594, which was last week's SpinRite story, which was about the RAID 10 that was having problems until the user ran SpinRite on his drives. He said: "I just wanted to note something I learned from an older and wiser geek."

And then Tim writes: "He told me that about half of the very substantial price difference between a commercial (Compaq at the time) server and something with similar specs that you built out with consumer parts was that Compaq has a large supply of hard drives that are not managed in a first-in/first-out manner." Meaning that Compaq maintains their own warehouse, essentially, of drives. And so he says: "Instead your server" - and he means here a Compaq server or any server - "would be assembled so that you did not have two drives from the same manufacturing run in your array because infant mortality in hard drives would tend to cluster around certain manufacturing runs."

Which is known. "If one drive failed early, and all drives were from the same manufacturing run, it's more likely that a second drive would fail before a new drive could be swapped in and the array rebuilt." And he finishes, saying: "This is the reason I have always purchased drives for my Drobos one at a time."

And I thought that was really interesting because there are famous examples. I remember, like, where a major drive manufacturer will go through a rough period, where they just sort of, you know, sometimes in the semiconductor fab industry it's called "losing the process," where something gets contaminated, or something happens, and they'll produce a big batch of drives, or in some cases processors, which pass QC, get through test, but there's a little whisper of a problem or something that only reveals itself out in the field. And those tend, because everything is done in a big production batch, and then the assembly line is switched over to make a bunch of something else, and then later it's switched back, there is a temporal and a sort of a batch-based connection.

And so what I liked about this was if you really did want to assure yourself of the most robust result, you would rather have a heterogeneous collection of drives than a homogeneous collection. And really you would not want to populate a large RAID with just all drives from a single batch because - if there was a problem. Now, the flipside is they might all be really, really good, rather than to have a problem. But in a RAID the whole point is to protect yourself from a low-probability but high-value failure. And spreading your drives around makes a lot of sense. So anyway, Tim, thank you for that tip. That's one that made total sense when I heard it, but it hadn't ever occurred to me independently.

So, What's Up With WhatsApp? Someday, hopefully, we'll have a widely recognized single understanding of the term "backdoor." We don't have that yet. The confusion arises partly because the term is good for baiting clicks on websites; and a backdoor is a scary term, partly because it seems sinister, and also because it's a brilliant term. Everyone can readily visualize a backdoor. As we saw during the FBI's request for a "we're not asking for a backdoor, we want a golden key to the front door" debacle last year, the proper use of the term requires an understanding of the intent.

A UC Berkeley security researcher, Tobias Boelter, waited nine months after informing Facebook and WhatsApp of what he believed to be a security vulnerability in WhatsApp's implementation of the Signal protocol. When he was ignored and obtained no satisfaction, he went public, and the Guardian dropped the bomb using the heavily freighted term "backdoor." The trouble was, and is, it was neither a mistake, which Tobias found, nor a secret. It's a deliberate and carefully considered design decision. This doesn't mean necessarily that it's the correct decision, but it does definitely mean that it's not a backdoor in the usual way that we still somewhat softly understand the term.

A backdoor is, I think, if we were to define it, by definition an unknown and secret password or something like that, a secret something, embedded for example into router firmware that allows anyone who knows the secret incantation, whatever it is, to obtain unofficial and unauthorized access to the device, whatever is being protected. That's a backdoor. But figuring out how a deliberate design decision feature can be abused, while it may not be a good thing, it's not a backdoor.

So shame on The Guardian for succumbing to the temptation to use that term. It achieved its goal, but I would presume at some small increment of reputation cost to them because there's been a lot of back push on this. Moxie posted, and he was really unhappy at Open Whisper Systems, I mean, and it caused a ruckus. Well, and no kidding because Facebook bought WhatsApp and its, as I said at the top of the show, trillion

messages a year. It is the most heavily used, supposedly secure, end-to-end secure instant messaging system, and even delayed messaging system, which actually is where the story goes, that exists, just because of the size of Facebook.

So what's this all about? It's another classic instance of a tradeoff between true security and security transparency. Users say they want encryption privacy. We know that privacy requires authentication to be meaningful. That is, you have to know who you're talking to. It's not enough just to have arranged a private communication. If you're talking to the man in the middle, then you don't have privacy. But edge cases arise that force a decision about whether to involve the user. As we discussed on Episode 555, and I remember at the time people saying, "Are you going to make this a special episode about the classic timer chip, the NE555?" I said, uh, no. Instead we talked about WhatsApp. That was the title of podcast #555, where we explained that WhatsApp is an implementation of Open Whisper Systems Signal protocol.

In its introduction back then, the introduction to the podcast, I summarized the news of the week and concluded: "And we will discuss the result of my deep dive into the Open Whisper Systems Signal communications protocol that's finally been fully integrated into the world's number one multiplatform messaging system, WhatsApp, along with" - and I said this at the time - "two things" - and what was it? That was 40 weeks ago. So if we're at Episode 595, that was 555. And back then I said: "...along with two things that must be done to get true security." And it's not entirely a consequence of the fact that they're not, but that would at least mitigate this and provide notification.

So what's going on? Performing real-time online messaging, where both endpoints are guaranteed to be online for the conversation, such as with any VPN, it turns out that's an easier problem to solve securely because, as we've discussed before, we have robust protocols, such as the Diffie-Hellman Key Agreement, which allow the two ends to securely negotiate a new shared secret, on the fly, in full view, with their traffic being monitored. Yet nobody in the middle, nobody passively examining the traffic can figure out what key it is that the two agreed upon, even seeing them talk. It's a perfect piece of crypto technology.

So long as reliable authentication is tied into the negotiation, meaning that you are sure who you're negotiating the new key with, to prevent tampering up to and including a man-in-the-middle attack, the endpoints are free to generate shared encryption keys at the start of every dialog and even to regenerate them periodically for long-lived communications. You wouldn't want to, for example, use that key forever if you had a VPN that was just up statically connecting two offices because that would mean you don't have what we call "perfect forward secrecy." If you'd always been using that key for a long period of time, and it became compromised in the future, then that key could be - and all of the historical traffic had been captured, as we believe the NSA is doing with their massive drive farm in Nevada - no, not Nevada, in Utah - then the problem is the future compromise of the key would allow all past communication to be decrypted.

So a more secure system is periodically expiring the key. During the ongoing traffic, one end will send to the other, hey, let's do another key agreement. And they'll both take new random numbers, while everything is going on agree on a new key, and then synchronize their change over the new key. So they're constantly expiring and switching to new keys so that at any point that a key got exposed somehow, the key probably in use, well, the key that was in use would only compromise the piece of traffic, the piece of conversation that was in place at the time. Okay, but that's, by definition, everything I just said assumes the endpoints are online together, and they can do this in real-time. That's effectively synchronous communications, where they're able to dynamically interact.

But the plot thickens a great deal, actually, when we ask for the same sorts of guarantees of authentication and secrecy, that is, privacy, from an asynchronous communication system like messaging. In the asynchronous scenario, we want to send a secure message to the other endpoint, even if it's not currently online. And we discussed 40 weeks ago how the Signal protocol arranges for this. And that's, of course, what WhatsApp inherits from Signal. Signal and WhatsApp ask clients to pregenerate a substantial block of 100 public key instances which will be escrowed on the central messaging server. So these keys are then distributed on an as-needed basis to anyone who wishes to obtain a key that is going to be a public key for the message's recipient. And that key would then be used to encrypt the message which is going to be sent to that recipient.

So if endpoints are offline, not only does the messaging server hold essentially prekeys, these keys available for distribution while a client is unavailable, to allow people to asynchronously encrypt messages; but, if that endpoint is offline, the messaging server must and does retain all unsent messages in encrypted form. It can't read them, so it's just storing blobs. But it's waiting for the recipient to come back, to log back into WhatsApp or Signal. And then it says, ah, and then it flushes those messages to it, and that client is then able to decrypt those messages because it's got the master key from which these ephemeral public keys were created.

So here's the problem. What happens when that other endpoint, that recipient endpoint, changes phones, or SIM cards, as is more frequently done outside the U.S., or when the application is reinstalled, while there are still pending messages encrypted for its previously keyed identity? So again, the scenario is we have an asynchronous messaging platform. All clients keep the central server loaded with keys to use in the future by other people who wish to send them an encrypted message that the middleman, the messaging server, nor anybody else listening, is able to decrypt.

So they sort of made a promise. They've set up a bunch of keys and said, "Use these to send me stuff." And then they go offline. Then a bunch of their contacts do send them things, but because they're offline the server holds them. Now, while staying offline, they change keys. Something causes them to change the key. WhatsApp gets reinstalled, they get a new phone, they change SIM cards, whatever, or just reinstalling WhatsApp. So now we have a problem. The root key which created these ephemeral public keys is different and will not be able to decrypt the messages that come back to it when it next connects.

So this is where Signal and WhatsApp diverged in their policies. Signal decided we're going to go for privacy first. We're not going to compromise that. So what happens is they block any additional communications and notify the sender that the recipient's key has changed and require affirmative acknowledgement and agreement. Now, and understand that this could be a malicious key change. So the key changing is one of the tricks in this system, and that's one of the things that we talked about at the time.

And I said to people, the thing that annoys me is that is off by default. That is, the notification of a contact's key changing is not on. So I told everybody listening, if you're going to use WhatsApp, and I don't remember what the case was with Signal, but certainly with WhatsApp, if you're going to use it, go into the security settings and turn that on because you absolutely want to know if a key changes. Not being notified...

Leo: Signal always tells you.

Steve: Okay. So there's no ability to disable that.

Leo: No.

Steve: Okay, good. And so, of course, that was one of the two things was make sure you had that turned on. The other is you could take it on faith that the key you're being displayed or that you've been given belongs to the person you think you're talking to. But authentication is incomplete unless - and there's really no way to automate that. We want to keep not taking responsibility for authentication, but it's ultimately up to us, if we really care. I'm not telling everyone they have to care.

But if you really do care, you need to authenticate, meaning that there is that block of signature digits that represents the other end's key, and they've got yours, and you need to somehow, hopefully out of channel, that is, don't use the messaging app itself to send that back and forth because that would be self-defeating. You need to use an out-of-channel means like talking to each other or emailing some of the digits or something. It's just a hash of your key, so it's not sensitive information, but it can be used to verify the key.

So Signal decided they're going to alert the user to a contact's key changing. Not only did WhatsApp decide not to alert, but they also decided, you know, we need to make this more transparent to the user. We get 15 trillion messages a year. This thing is too popular. We just don't want to, like...

Leo: Rock the boat?

Steve: Exactly, rub people's face around in something messy because they won't like it. They won't understand it. They won't know what is happening. So what WhatsApp does is the way the protocol works, the sender of the message that posted the message onto the server holds it until the server sends back a notification that that held message has been received successfully by its recipient. And as I was reading this, there's apparently a one check and a double check. And so the one check is the server got it, and the double check showing is the recipient got it. So while there's a one check on the sent messages, your client still has them.

And so again, purely for the sake of convenience, WhatsApp chose to make the key change absolutely transparent. The way they do that is that when the recipient reconnects with a new key, WhatsApp realizes, the WhatsApp server realizes that and says, oh, shoot. I've got a bunch of messages for you, pending, encrypted under your old set of public keys that you can no longer access because you've just changed your master key. So hold on a second.

The messaging server obtains a new blob of 100 ephemeral keys for that newly rekeyed endpoint, redistributes them to all the clients whose messages are pending and says, we had a problem in transmission. Would you please reencrypt those messages you sent before, but haven't deleted because we told you not to, because we hadn't confirmed their delivery. We need you to reencrypt them for us under this new key. Without any fanfare, no user notification, nothing happens, that all happens behind the scenes.

So the messages that had only kind of been half sent get resent under the new recipient's key. And they go back to the messaging server. Now they can be sent to the

newly rekeyed recipient, who can decrypt them. And that notification goes back to the messaging server that sends it back to the senders, and they get the double check on the messages, and everybody's happy. Well, not everybody because it's clear this - it's not a backdoor, it's a problem, if you absolutely really care about security. The problem is now that everybody knows about this. I mean, maybe some people knew about it. This is why I believe that disclosure is always the best policy.

But all law enforcement in the world now knows that it is possible to cause the most secure messaging platform on the planet - the most popular, not the most secure, the most popular supposedly secure messaging platform on the planet - to not notify endpoints of message receipt, that is, not give them the second check and permission to delete it locally, and to give them any key they want and cause the messages which have not yet been sent to be resent under a key that a third party could decrypt. And it turns out that it doesn't - the argument has been, oh, that only applies to the most recent message. No. If the protocol is obeyed, that's true. But nothing enforces the protocol.

Tobias demonstrated that it is possible to have a conversation back and forth between endpoints where the messages are never being confirmed. The conversations happen, but the final message is not sent from the messaging server, preventing the endpoint from dispensing the messages. And in real-time there can be constant rekeying going on which is completely transparent to the user. So we don't know this has ever happened. We don't have any reason to believe it would be. But this is a consequence of the decision that WhatsApp made not to give people a dialogue box they won't understand. Just like, wait, wait, what do you mean, my contact's key changed? I mean, what? What are you going to do with that information?

And so they said, no, we're just, you know, not only - oh, and even if they did turn on the security switch, saying I want to be notified of any change to my contact keys, it turns out all pending messages are rekeyed and re-sent before the dialogue is displayed. Well, figure - explain. Figure that, well, you know, why? But that's what happens. That's the way the system is designed. So that's what this was. I love this because it's a perfect story for this podcast. Moxie and Open Whisper Systems understood the problem, I mean, and solved this difficult asynchronous messaging system in a clever way, with this notion of a blob of prekeys that are used ephemerally in order to allow offline encrypted messages to be stored by a central server that cannot decrypt them.

But then you have the problem - if you're in real-time, there's no problem with a key being changed when you're offline because you're online. But if you're asynchronous, either end's key could change and does change for any number of reasons. So they had to figure out how to do that. They did that by causing the endpoints to hold unconfirmed received messages until they had been confirmed received, and the ability to have them rekeyed and resent. And with that goes notifying the user.

But, and as you said, Signal does that. Unfortunately, as you start chipping away at the details and take away notification and then make it completely transparent, and even if you notify, you do it after the fact? Okay, well, then you've really got broken privacy. Again, we don't know that it's going to happen. But now what we're seeing is law enforcement is being very aggressive about what they're asking for, and they're getting courts to ask for what they want, and they're compelling companies to comply.

So this is an interesting tradeoff. I read Moxie's complete rebuttal. He was standing up for Signal and WhatsApp. And he argues that, okay, yes, the way The Guardian portrayed it as a backdoor was clearly wrong, and I completely agree; that WhatsApp had a hard decision to make. Were they going to offer privacy, but set the bar to a point that it would be a problem for users, which might drive them to, as Moxie put it, some

other less secure solution? That is, this is good, but unfortunately it can be manipulated because of the details of the way it was designed.

So I'm still with Threema. I don't actually use Threema because I don't have anything I'm sending my friends except, oh, did you know that Season 6 of Homeland has restarted? It's like, okay. But here's my message to anybody who is paying more than [audio anomaly]. If you actually truly really do need security, then you're going to have to sacrifice some of this convenience. And I would argue most people on Facebook, they're also, you know, retexting that Season 6 of Homeland has started. They're not doing anything more that requires real protection from law enforcement or authoritarian governments and so forth. If you do, then what I love about Threema is that it doesn't have any of this complexity. It is dirt simple. Which is why it's trustable. It's now been audited, too.

And remember, with Threema, it does nothing for you. It's 100% up to you to verify the key of the person at the other end. And they provide means to do that. But it's up to you to do it. And if the other end changes their key, you've got a broken link. But wouldn't you rather have a broken link than have an unknown link to some foreign government, or your own government? So that's the tradeoff.

Anyway, I love this for the podcast because it's a perfect instance of the press mischaracterizing this, but there being a true dilemma because we're, despite everything Moxie and company did, this is the best they could come up with. And it's very good. But you then say, oh, yeah, it's very good, but we can't be, like, popping up some scary thing every time a key changes. Okay? But part of the security model was that we would do that. And so if you start chipping away at it, then you do open this kind of, it's not a backdoor, but it is an opportunity for exploitation or for someone to be compelled to use the system in a way the protocol was not designed to be used. And it will do that. You need no modification to the clients, only the central server. And you can create a third-party tap.

Leo: So action items would be use Signal.

Steve: Yes.

Leo: Or turn on the warning in WhatsApp, in the settings.

Steve: Yes.

Leo: Then it would be roughly equivalent to Signal, or would it be exactly equivalent to Signal?

Steve: Not exactly, because unfortunately it notifies after the horses have left the barn.

Leo: After, right, okay.

Steve: So when you were sending your secret location of the rendezvous to meet with

the other bad people, that message got out and could get decrypted. And unfortunately, only then it pops it up. But by that point, unfortunately, the decryption is done.

Leo: If you're really up to no good, I would think that the best thing to do would be to install PGP, generate long keys, 4,096-bit keys, be very careful about protecting your private key, and then just using - because it's easy to encrypt with PGP anything; right? Or are you saying a one-time pad? Or just a pen and paper? That's what President Trump says.

Steve: Unfortunately, I did see that, as a matter of fact.

Leo: Yeah, write a message and send it with a courier.

Steve: It is hard to argue. And in fact we've talked about it. We've said, "Go into the middle of a park and whisper to each other."

Leo: That's better than paper, yeah.

Steve: "After you throw a coat over both of your heads so there's no audio leakage and no lip reading being done." I mean, it really is a problem. And remember that we still have the problem that, if some malware, or we now have a word for that, it's called an "implant," that's the official NSA term, an implant in the phone will still be sending stuff out before it gets encrypted. So, yeah.

Leo: You might have noted the use of Telegram in the dossier. And in fact the dossier implied that Telegram had been hacked. And Telegram was at great pains saying, no, no, we weren't hacked. Of course we don't know if the dossier is real or not, or if all the items are real, or some are real, whatever. But I have to say, you know, I remember I told you there was a dog whistle in the blog post, Telegram blog post a few months ago, where they had Pepe in there, Pepe the Frog. And remember it's written by a Russian, Pavel Durov, who claims to hate Russia because they stole, you know, he was the Zuckerberg of Russia. They stole his Facebook clone. Putin's government forced him to sell [crosstalk] fire sale, and so he's fled Russia. Yeah, but maybe not. And I don't know. I wouldn't - we've talked about the fact that Telegram uses its own roll-your-own crypto that isn't really necessarily safe. We certainly don't know how safe it is. So don't use Telegram. Use Signal. Or Threema. Now, Threema's not open source, but you say it's been audited.

Steve: Yes.

Leo: So that's as good. Okay. Just, you know, in case you're up to no good.

Steve: I use iMessage. I use iMessage, and I drive my car around, and I don't worry about any [crosstalk].

Leo: iMessage is no better than a lot of other systems where the key is held by the company.

Steve: Correct. Key management, remote key management is the bugaboo.

Leo: Yeah. Because that means Apple has access to the messages.

Steve: Yeah. If you're not authenticating yourself, then you're not secure. I mean, you probably are. But you're not guaranteed. There's no way, we have not found a way of subcontracting authentication.

Leo: Yeah. Yeah, it's an interesting, interesting world.

Steve: It's an interesting theoretical problem.

Leo: It is, it's more interesting from a theoretical point of view, exactly.

Steve: Yeah.

Leo: Yeah. Steve does this show every Tuesday at about 1:30 Pacific, 4:30 Eastern, 21:30 UTC. If you'd like to stop by and watch live, join us in the chatroom at irc.twit.tv, we'd love to have you. But you don't have to because we have on-demand versions of the show. Steve's got them at his site, GRC.com. That's one place you can get them, GRC.com. He also has transcripts. That's the only place you can get that. He also has SpinRite, the world's best hard drive recovery and maintenance utility, his bread and butter. GRC.com. Lots of other good stuff there, as well.

We store audio and video at our website, TWiT.tv/sn. You can watch live there or watch live on YouTube, YouTube.com/twit. There's also a Security Now! channel on YouTube. You can watch the video there if you want. And you can download copies of audio and video from anywhere you get your podcasts, including iTunes and Google. Google Play Music has our show. Spotify has our show. Stitcher. Slacker has our show. TuneIn has our show. It's everywhere. Just subscribe. That way you won't miss an episode. Thanks for joining us. Next week you want to do questions?

Steve: Let's try, yes.

Leo: Leave your questions for Steve. He's @SGgrc on the Twitter. Or you can go to GRC.com/feedback and fill out the feedback form there.

Steve: Yup.

Leo: And we will get to as many as we can next week, good lord willing and the creeks don't rise. Thank you, Steve.

Steve: Thank you, my friend. Talk to you next week.