iOS Security Part 3

Description: (Part 3 of 3) On the heels of Apple's major update to their iOS Security whitepaper, Steve and Leo catch up with the week's top security news - one IMPORTANT Microsoft Zero-Day Fixit, but otherwise largely debunking a bunch of hysterical headlines and "news" stories. Then they FINALLY conclude what has become the three-part series describing the security of iOS v7. Unfortunately, this week the news is less good.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-448.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-448-lq.mp3

SHOW TEASE: Time for Security Now!. Steve wraps up his look at iOS Security, and here comes the bad news. Plus bad news about a zero-day flaw in Microsoft Office. It's all next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 448, recorded March 25th, 2014: iOS Security Part 3.

It's time for Security Now!, the show that protects you and your privacy online, with Mr. Steven Gibson of GRC.com. Seems like we were just together moments ago. Oh, we were.

Steve Gibson: Twenty-four hours ago, Leo, yes.

Leo: Very kindly, Steve spent some time with us on Triangulation yesterday. If you haven't seen that interview, the nice thing about getting our hosts on Triangulation is the chance - and we've done almost everybody now.

Steve: Oh, good.

Leo: Yeah, it's a chance to spend an hour talking about the things we don't talk about on the air. So we talked about your youth, that great picture of you as a four-year-old doing some wiring, your influences, and your advertising agency. Still cracks me up.

Steve: Somebody, I got a tweet from someone saying they're still laughing about the name of my...

Leo: Gibson & Garnish.

Steve: Yup.

Leo: Find out more at Triangulation, TWiT.tv/tri. Now, what are we doing this week on Security Now!, Mr. G?

Steve: The never-ending series on Apple iOS Security.

Leo: Will finally end. Yes?

Steve: It will finally end.

Leo: Oh, good.

Steve: Unfortunately, on a bit of a sour note. Essentially, and this really, I mean, this is a great lesson for us because one of the overarching themes that we have seen develop through the industry and reflected in the podcast, is that security is difficult, and that it's very hard to get these things right. So I assume we're going to have time to get to the very last thing I want to talk about, which is what is jailbreaking and how can you have jailbreaking if you have everything we've talked about so far.

But there are two things specifically that we haven't talked about, we just haven't gotten to because there was so much I wanted to cover. And I really wanted to, as we have in the last two podcasts, lay down a good understanding and foundation of what it is that Apple has created in the iPhone/iPad/iOS platform. And we'll talk about iMessage and how, unfortunately, what they say is not true about iMessage; and about putting your keys in the cloud, which is the most worrisome thing of all. And of course we've got a catch-up with the news. This week, it must have been a slow news week for the industry because most of what I have to talk about is debunking hysterical headlines.

Leo: Well, I think you're going to be doing a lot more of that over the years because just people - sensationalism garners clicks, frankly.

Steve: Well, and even in a story picked up from a respected, I assumed, journal about Internet security claimed that WPA2 WiFi had been cracked. And everyone picked up on it and echoed this absolute nonsense. Like, nothing happened.

Leo: You know, it's a nexus between sensationalism and a lack of technical depth or understanding or context, maybe, would be a better word.

Steve: Yes.

Leo: Paranoia. And you mix those up. I'm reminded, remember the Pwn2Own, the guy who does Pwn2Own, which is a great idea, wrote the article about how a virus had jumped the air gap between his computer and another computer. And everybody got all up in arms. And then, you know, this is a reliable guy, one thinks.

Steve: Yeah, he had some credibility that he had built up over time. And he was absolutely sure that this virus was jumping across breeds of computers. And many people who actually understand where BIOSes come from explained how that's absolutely not possible. I mean, it's like the Slammer worm infecting your cat. I mean, it's like, no, it's not...

Leo: Different species.

Steve: It's not that kind of worm.

Leo: But I think that partly also it's a much bigger topic than it used to be, and no one person grasps the whole of it. So it's easy for somebody who does grasp some of it to miss another piece of it. Apple was an example here, too.

Steve: Yes. And in this instance, the article - this journal is an academic journal behind a paywall. And when I looked in all the standard places to find somebody who had, like, innocently let go of it, I paid my 30 euros to purchase it. And so one of the other problems may be that it just wasn't generally available. In, like, tracking this down, one news source did this sensationalized treatment. And looking carefully at the wording in everybody else who also said the same thing, it looked like they were all copying from this single source. So that's another problem we have.

Leo: On the bright side, there is a process, you have to be patient, over time that this stuff can - I think in the long run the truth outs.

Steve: It's called the Security Now! podcast.

Leo: Yeah. People listen to this show, and they go, okay, let's get the truth here.

Steve: That's what we're here for. We'll find out what's going on. So we're going to talk about an important, the one really important thing is just a couple days ago, actually I guess it was yesterday, Microsoft released the news that they were aware of a zero-day vulnerability in Word which does require some immediate action from our listeners. There's a Fixit click quickie before they do the official patch. We'll talk about the enhanced mitigation experience toolkit, EMET, which we've never spoken of before, though it's been around for years, about this ridiculous claim that WPA2 has been cracked.

Also, again, even from someone as venerable as Symantec, there's, like, hysterical headlines saying that you can text an ATM to give you money. Then Google has announced some improvements to Gmail. I've seen something that is what I consider the very first clear overstep of Snowden's charter, which is worrisome. And a new version of Firefox and some other updates and things. So a great podcast for us.

Leo: Let's get the security news before we dive into this iOS Security stuff.

Steve: Yeah. So the most important thing is that what was discovered by Microsoft, and we don't really know too much about the history of this, but they are saying that this is being exploited in the wild. Due to the nature of it, it's probably targeted attacks. So these are probably bad guys that found a way to get their code into a person's computer who are sending them specially crafted emails.

Leo: Spearphishing.

Steve: Exactly. The entry point is the RTF format renderer for Microsoft Word, Office's version 2003, 2007, 2010, and 2013. RTF is the Rich Text Format which is - it's been around for, wow, decades, I guess.

Leo: Oh, I always use it. I mean, it was created by Microsoft, but it's kind of the lingua franca of word processors.

Steve: Yeah. And it predates HTML. The normal way you have of viewing RTF files in a Windows system is WordPad. Notepad is like the raw text file viewer and editor. WordPad, what is different about WordPad is that it's essentially just a user interfaced container around the RTF renderer. So it shows RTF files, Rich Text Format files. And naturally, with Word, Word is sort of intended to be a superset of that. It of course understands DOC and DOCX, but also all the way back. Obviously you can open a TXT file; you can open an RTF file.

Leo: Yeah. Well, and RTF I should say is used also by Evernote. That's its format for the notes you store in Evernote. And it is also used by Outlook and many other email programs as one of the alternatives to HTML mail or plain text. In fact, most of the clients that I use offer RTF, and sometimes even by default, as their formatting.

Steve: Right, you're able to make embellished text, which is...

Leo: Yeah, bold fonts, typefaces, you know.

Steve: Right, exactly. Color, bold, font size, font change, and so forth.

Leo: I thought it was just a pure data format. I didn't think it was dangerous.

Steve: Well, and this is the problem, is that essentially there's always going to be an interpreter which is reading the text file. And, for example, in RTF, you have escape events where you say, like you have a font change escape, where it's backslash and then a command and then some parameters for the font. Which means that there's code that gets involved in interpreting that escape sequence, and then looking up and loading and then rendering in that font. So if there's a mistake in that interpretation, that presents an opportunity for a vulnerability. And that's what someone found.

So the normal vector is Word. But Microsoft notes that the same RTF renderer is also the default renderer for Outlook. So not only opening a Word document which is in RTF format, but even email as a vector is a way for this thing to get in. So they note that there are mitigations in Office 2013. Even though the problem is still there, that is, the RTF renderer for the latest version of Office, Office 2013, still has the problem. But, and we'll talk about this in a minute, there are mitigations that prevent it from succeeding, if that's the version of Office you have.

So anyway, so that's the problem. It's a newly discovered means of executing code through a deliberately malformed file, in this case an RTF format file. And in their TechNet blog, Microsoft said: "The in-the-wild exploit takes advantage of an unspecified" - which means Microsoft is not specifying it. They're wanting to keep this quiet. An unspecified, I mean, they know what it is, "an unspecified RTF parsing vulnerability combined with an ASLR bypass." I mention that because we've just been talking about Address Space Layout Randomization and how useful that is in mitigating the damage which something can do if there's a vulnerability that's exploited by making it harder for the bad guys to know where things are in your system. Through randomizing the layout of the address space, you're often able to completely prevent this bad thing from happening.

So there is an ASLR bypass which this thing has worked out, which depends on a module loaded at a predictable memory address. They said: "First, our tests showed that EMET default configuration can block the exploits seen in the wild. In this case, EMET's mitigations, such as mandatory ASLR and anti-ROP" - we've talked about that also just recently in the context of the iOS podcasts. That's the return-to-code exploits, where something knows where some code is. It's unable to execute code itself because, for example, it just - there may be restrictions of one form or another, the amount of buffer space it has, that is, that the exploit has, or the nature of the constraints of the execution environment. But it's able to jump to something located, for example, at a known location in the operating system and execute that code. So this is the anti-ROP features. And Microsoft says those things in EMET effectively stop the exploit. Then they say, "You can find out more information about EMET at..." and it's just Microsoft.com/emet. For anyone who's wondering, we'll talk about that in a second.

So this is bad enough that some people in the industry have speculated that Microsoft may do an out-of-cycle patch. We're two weeks way from this next second Tuesday of the month, which will be famously the last gasp of XP's updates, April 8th. They may or may not have it in time for that. But there is a Fixit for this. I don't have - I didn't do a quick link, unfortunately. It just skipped my memory, or my mind. I did tweet it this morning, some news of this. So it's definitely worth doing. Anyone who, well, anyone who's using Windows and has Office installed should consider this, if anything you're doing might cause you to encounter a document or a file or even email that could be malicious. And we don't know how quickly this is going to scale up. We don't have a sense for how widespread this is. But it's real, and definitely worth taking a look at.

Now, this Enhanced Mitigation Experience Toolkit is interesting. It's been floating around for many years' worth of podcast, and I've just never gotten around to discussing it. One

of the reasons is that it is not easily usable. It's not the kind of thing where, for example, like an AV system, where you just pretty much drop it in, and you never have to think about it, and it takes responsibility for keeping you safe. What this Enhanced Mitigation Experience Toolkit does is it forces on a number of well-known mitigations that we have talked about. We've talked about all the pieces of this. The problem is because Microsoft got such an early start on building these systems; whereas, for example, iOS had the advantage of starting much later. Apple was able to say, with the iPhone, which is very new compared to the beginning of Windows, which is now decades old, we're going to incorporate these things from the beginning.

And this is the problem. It turns out, if you enforce Address Space Layout Randomization, which is a powerful mitigation, many things break. So Microsoft is stuck. They would like to turn this on, but they just can't because it breaks too many things. And the reason it breaks them is it wasn't always there. And this is the lesson of all of this is, for example, Apple is able to always enforce Address Space Layout Randomization because the concept existed before iOS did. And it was proven and clearly a good thing.

So as I mentioned last week, as part of the development environment and the general iOS environment, the Xcode system for creating iOS apps always has that on. And the point is therefore, as a developer, you are never able to depend upon in any way the known position of something. Whereas, if you developed an environment which never had it on, you might make some assumptions about that environment which turning it subsequently on would break.

So this Enhanced Mitigation Experience Toolkit is free. It's at 4.1 now, with a technical preview of 5.0 coming. I mention it now because it's one additional thing that people who wish to continue to using XP after April 8th might add to their toolkit. Again, I would say, I did say last week, don't use Office. And this, the problem we've just been speaking about, this is not a problem with XP. This is a problem with the stuff that contacts the environment, like your browser, like Office, like email and so forth. So that's why I was saying Libre Office made more sense than Office, if you wanted something that's going to continue to receive care. But the reason I say this with caution, that is, not just for everyone to go install it, is from Microsoft's own statement, they said: "The Enhanced Mitigation Experience Toolkit is a utility" - and I love the name "Enhanced Mitigation Experience." It's like, okay, well...

Leo: What the hell?

Steve: I know, "...is a utility that helps prevent vulnerabilities in software from being successfully exploited." So again, this is where we made it very clear last week, and there are two things. There's can the software somehow get a foothold, that is, malware, something malicious, the bad guy get a foothold; and, if it can, what can it do about it? And so the mitigation is to mitigate the damage done from a vulnerability that's discovered. So Microsoft says: "EMET achieves this goal using security mitigation technologies. These technologies function as special protections and obstacles that an exploit author must defeat to exploit software vulnerabilities." That's perfectly clear. "These security mitigation technologies do not guarantee that vulnerabilities cannot be exploited. However, they work to make exploitation as difficult as possible to perform." True, if those mitigations are present. And they're not normally because, unfortunately, they break too many things.

So why not just - I just answered my own question, why not just build this in? Even Microsoft said: "When EMET mitigations are applied to certain software or certain kinds of

software, compatibility issues may occur" - and this is one of those "may occur," where we know what they really mean - "because the protected software behaves similarly to how an exploit would behave." So this is - it is in a sense behavior monitoring. It's looking at the behavior of the software and saying, eh, we don't think you should be doing that. But it's possible that good software might want to do the same thing.

Then here's where I got a big kick out of this because they said - this is, again, Microsoft's own explanation, right upfront, of EMET: "The following is a list of special products" - I'm sorry - "of specific products" - just to give you a flavor, a sense - "that have shown compatibility issues with the mitigations that are offered by EMET. You will have to disable specific incompatible mitigations if you want to protect the product by using EMET." Which sort of defeats the purpose.

Leo: Yeah, yeah.

Steve: "Be aware that the list takes into consideration default settings for the product. Compatibility issues may be introduced" - that is, above and beyond - "when you install certain add-ins or additional components to the standard software." So as a sample, the very popular 7-Zip Console GUI and File Manager; Adobe Acrobat; Acrobat Reader; certain AMD/ATI video drivers; Apple iTunes; Dropbox; Google Chrome, of all things; Google Talk; Oracle's Java, well, that might be a blessing, actually. But not only other people's things. Even Skype doesn't work with EMET on. Nor does Windows Media Player or Windows Live Photo Gallery. So you get a sense for the fact that anyone who's going to choose to have the enhanced mitigation experience needs to also take responsibility for all the things it's going to break and then go in and spend some time tuning and tweaking in order to get things that used to work before you enhanced your mitigation experience in order to bring it back to the ground and make it work.

So it's there. It's one more tool, a useful tool in the toolbox. And it may well be that in constrained environments where people for whatever reason really do want to continue using XP, we already know you shouldn't use IE. You should not run as an admin. Oh, and by the way, not running as an administrator prevents this whole zero-day exploit.

Leo: Ah.

Steve: Yeah, so there again, the same advice, make yourself a disempowered user, not admin, and you don't have any of this problem anyway. So that's that news. Definitely worth disabling this. I'm sure Microsoft will be patching it. I imagine that they ought to have time in the next two weeks to get this done by the deadline.

Leo: Do they have to find a new renderer for RTF or...

Steve: No, no, no. They just have to remove a jump instruction or something. I mean, it's like it's some - it's something where they did a string copy and forgot to check the size of the buffer of the destination, and it turns out you could maliciously - you would give it a font name that's 4K long, and it would just wipe out something inside of the kernel and give the attacker control, that kind of thing. And they stick their shell code as part of the font name, and then the routine that called it would jump into the code rather than into its own return instructions, execute your code, and then off you go. That kind

of thing. So probably a buffer overrun somewhere in the RTF renderer. So it's like, ooh, we only check the size of the destination buffer. Yeah, uh-huh. And if you'd only not put GDI in the kernel...

Leo: It would be easier, wouldn't it.

Steve: Oh, god, none of this, yes. This was done because, when I was ranting at some point recently, I don't know if it was yesterday on Triangulation or in the podcast the week or two before, but I was talking about how Microsoft was always - oh, it was yesterday because we were talking about craftsmanship in software construction and how - remember how slow Windows was in the old days? I mean, I only started, I only went into Windows in order to run Micrographics Designer, that was this beautiful early drawing tool. Otherwise, I just got out of there as quickly as I could because it was just painful to use. And then, thank god, we got graphics accelerators, the very first hardware blitters on display cards so that the blitting didn't have to be in software.

And to their credit, Microsoft did everything they could. They were just asking for more than the hardware could do, more than the hardware could give them. And at one point they decided, okay, there's so much communication between GDI, which was a module running in user space, down into the kernel, and these so-called "ring transitions," which is a security check. I mean, the ring transition is there because you don't want the unprivileged code to have privileges. So when you cross that boundary, there's a whole bunch of things, changes, that allows the code now to do things with privilege it didn't have before it crossed. That is time-consuming on an Intel system.

So what Microsoft decided was, oh, I know how we can make this faster. Let's take this whole blob, this GDI, the Graphics Device Interface, and put it in the kernel. Then it'll all be privileged, and there'll be no more ring transitions while it talks to the rest of the kernel.

Leo: And just maybe crash the machine like nobody's business.

Steve: Yes. And now all of your buffer overruns are in the kernel, rather than where they used to be, in userland, where they couldn't do nearly as much damage as they could. So we've been paying that price for that decision now for a decade.

Leo: You saw the - maybe you didn't - the Android exploit that can force reboots of your Android 4 and later by having a 387,000-character application name. And, you know, I guess they never really thought that anybody would do that, or something.

Steve: If there's a will...

Leo: Who found that?

Steve: Ow. Well, and you know what it was, it's one of the benefits of open source, which, again, it's a mixed blessing.

Leo: You know where the errors are.

Steve: White hats can look at it and go, oh, wait, look at this. Here's a string copy that's not checking the size of its destination buffer. Unfortunately, bad guys can look at that and go, oh, wait, look, here's a string copy that's not checking the size of its dest- oh, look. They're copying the application name. Let's give it a monster. And, pow.

Leo: Boom.

Steve: Yeah. Okay. So WPA2 WiFi security cracked.

Leo: Oh, my god.

Steve: Leo, that's what it says.

Leo: Breaking news.

Steve: On many, many Internet - and we know that everything on the Internet...

Leo: It's all true.

Steve: ...is true.

Leo: All true.

Steve: Yeah. And so if you google right now "WPA2 WiFi Security Cracked," Google will obligingly give you links to all of these articles that pretty much say the same thing because it all apparently came from the same place. It looks like it came from ScienceSpot.co.uk, Science News with Inderscience Research Spot, and so it was called Science Spot, where the headline was exactly as I said, "WPA2 Wireless Security Cracked." And unfortunately I cannot pronounce the names of these three guys. My guess is they had a paper due. And so they thought, okay, let's do a paper.

So these three guys at Brunel University in the U.K., the University of Macedonia in Greece, and I guess Lancaster University in the U.K., have investigated the vulnerabilities in WPA2 - this is what the article says - "and present its weakness. They say that this wireless security system might now be breached with relative ease by a malicious attack on a network." Oh, my god, that's headline news.

Leo: Wow.

Steve: I know. "They suggest that it is now a matter of urgency that security experts and programmers work together" - okay, we're going to work together, Leo, to fix this - "to remove the vulnerabilities in WPA2" - those pesky vulnerabilities - "in order to bolster its security or to develop alternative protocols" - maybe just scrap it, you know, we need WPA3 - "to keep our wireless networks safe from hackers and malware." And finally, they said, "The researchers" - whose names we cannot pronounce - "have now shown that a brute-force attack on the WPA2 password is possible, and that it can be exploited, although the time taken to break into a system rises with longer and longer passwords."

Leo: How much does it rise by, Steve?

Steve: Wow. You think? So this, verbatim, made headlines all over. ScienceDaily.com breathlessly repeated it.

Leo: What's funny is, if you do that search, you'll see articles from 2010, 2012, 2013 saying the same thing because this is nothing new. We've known this forever.

Steve: No, no. So I looked around, tried to find this anywhere. Sometimes - and this is clearly behind a paywall. Sometimes, though, you can just - other sites will just post them. So if you really have low resistance to paying, you just pay. If your resistance is a little higher, you poke around a bit, and you can often find the same thing. I find that when I'm...

Leo: Well, 30 euros is nontrivial. I mean, that's expensive.

Steve: Yeah, yeah. But thankfully, our listeners have been wonderful about purchasing SpinRite to recover their data, and in some cases just to say thanks for the podcast, or to purchase 6.0 because they know that 6.1 will run on their Mac. And so I can afford 30 euros for the sake of the podcast.

Leo: Thank you, Steve.

Steve: So in my effort to give back, I plunked down my money, I got this 12-page PDF and plowed into it, looking for what they had discovered. Now, they said that they had 10 test passwords. They used off-the-shelf apps, Linux-based things, AirSnort and Aircrack and the things we've talked about through the years. Nothing new. Nothing invented. This, I mean, and there's lots of citations. They have three pages of citations at the end of other people that they're talking about as they sort of give us the history of wireless WAN insecurities and all these problems. And they show 10 passwords. The first one was "icecream." The second one was "transubstantiation." And the other eight were gobbledygook.

Leo: Okay.

Steve: And in quoting them, they said: "In some of the cases the key was very simple,

Case 1 and 2."

Leo: Yeah, called "dictionary words."

Steve: "Whereas, in the other ones, the key was too complex."

Leo: Too complex.

Steve: And they say cases 3 through 10. Then they said, okay, in their discussion after this: "WPA/WPA2 are considered amongst the most secure protocols. This is due to the fact that, even having an instance of the preshared key, it requires a dictionary attack to break it." Now, just to remind our listeners, we talked about this years ago.

Leo: 2008, as a matter of fact.

Steve: Yes. Every time it comes up. And remember that, when we did abandon WEP, that was really badly broken [SN-011], which I think was the title of one of our podcasts, or even more badly broken than you know [SN-089]. And real, finally, security experts were involved in the design of WPA and WPA2. What they did was to mix the user's key with the SSID, essentially the router's broadcast identity, and they used PBKDF2 and 10,000 iterations of a hash function specifically because they understood, if you captured over the air, you captured some of the packets, you could then brute-force by guessing possible passwords. You'd have to mix in the SSID. That was the brilliance of doing that, which is why our advice back then was don't leave it set to Netgear or D-Link or Belkin or whatever. Make it your own. Doesn't really matter what you make it because you don't get any security from that except that that breaks any precomputation attacks where, like, for example, somebody could actually do all of this 10,000 iterations for a whole bunch of English common passwords against a given router's default SSID. And that would make the attack faster.

So this attack is huge. In WPA/WPA2, the brute-force attack is hugely slowed down, which is why our advice. And I think this might have been the genesis of the Perfect Passwords page, which I said just get some gibberish. Just use gibberish, and you're done. You just don't have a problem. However, these guys allege otherwise.

So they said: "The more complex the password is, the safer the network security will be. More precisely, words like icecream, computer, clouds, wireless, mynet, airhouse, et cetera, are commonly used, increasing the probability of finding the key in a short period of time. On the other hand, if the key consists of different types of characters - a combination of lowercase, uppercase, special characters, and numbers - the complexity would be increased. Hence, the adversary must have a dictionary consisting of all the different combinations of all the printable ASCII characters of all the possible lengths." Imagine that, Leo.

Leo: I wonder how big that would be.

Steve: Groundbreaking. "...[I]n order to ensure that" - and we have here a "he" with a

"(s)" in front of it, so we're being sexually neutral - "he or she will be able to find the secret key. In order to have a complete dictionary with all the different combinations of all the standard printable characters, the length [number of records] of the dictionary will be..." and then they have a very impressive-looking equation with summations and exponentials and things, which I actually think is wrong, but it's there.

Leo: This is definitely an undergraduate paper.

Steve: Ugh. Well, I don't get how it got published in the International Journal of Internet Security. It's like, what? Anyway, so then they run their equation. "By performing the calculations" - now, this is in their paper, Leo. "By performing the calculations, the complete dictionary would consist of 3.991929703310227 times 10^124 records. Thus," I'm quoting, "Thus this procedure (that creates and searches the dictionary) will last several weeks using a simple computer, due to the required time, which will be extremely high." Now, I thought, okay. Several weeks. We take 3.99 times 10^124. We'll round it up to 4 just make things easier. If you take - so that's the size of the dictionary. You take 60 seconds per minute.

Oh, I'm sorry. First you want to - how many can you do in a second? Let's be generous. We'll give them four, oh, no, we'll give them a billion. You can do a billion a second. You can't because of this 10,000-iteration hash. But for the sake of argument, you can test a billion in a second. So that's 10^9. We subtract that from 124 because the other one was 10^124. Now we're at about 4 times 10^115. Whittled it down substantially [clearing throat]. Now, 60 seconds in a minute; 60 minutes in an hour; 24 hours in a day; and 365.25 days per year, counting for leap year. We divide that into this 4 times 10^115 to give us the number of years it will take. Now we're down to only 1.264 times 10^108.

Leo: That's more than a few weeks.

Steve: That "few weeks" may qualify as the understatement that we have actually ever encountered on the podcast.

Leo: This is bizarre because it's so, just, wrong. It's wrong.

Steve: It's crazy. I mean, it just, it is. And, I mean, I don't understand why they wrote this. I don't understand why a real journal published this. And clearly somebody was scanning through the journal, or maybe the journal put out a press release. That's probably what happened. It's just like, whoa. And it was picked up by various people and given the headlines that WPA2 was cracked. It was very often tweeted to me, as you might, well, yup, there's Science Spot: "WPA2 Wireless Security Cracked." So I know that our listeners listen to get a...

Leo: Don't give these guys a doctorate, or whatever the hell they're trying to get. But the International Journal of Information and Computer Security does, I mean, I don't know that journal, but it does sound like a serious enterprise.

Steve: They got my money.

Leo: Yeah, 30 euros all to themselves.

Steve: Anyway, but I had to know what was going on, and nobody was publishing any details. So now we've got the details, and everyone can relax. I mean, WPA2 is very well vetted now. It has had the crap pounded out of it, and nothing has happened. So, I mean, we've talked about the various disassociation hacks and games you can play with ARP spoofing in an environment, I mean, there are high-end things. But fundamentally, it's solid. And the only weakness is that you would use a - you would leave the default SSID, which could provide an opportunity, if people developed dictionaries which would allow them not to have to do a custom dictionary for your particular SSID. And you've got to use a good password. You do that, and you're really safe.

Leo: Just unbelievable.

Steve: Now, not to be outdone, again because this apparently was a slow news week, Symantec, who we know knows better, their headline was, "Texting ATMs for Cash Shows Cybercriminals' Increasing Sophistication." And I thought, oh, no. And of course let's roll this into the end of XP's support IV drip from Microsoft, why don't we, even though it has nothing to do with it. So Symantec's blog posting starts: "There is a growing chorus of voices calling for businesses and home users to upgrade existing Windows XP installations to newer versions of Windows, if not for the features, then at least for the improved security and support. ATMs" - because we've got to mix this all together in this stew - "are basically computers [uh-huh] that control access to cash" - thus why the burglars are hovering around them.

"As it turns out, almost 95% of them run on versions [gasp] of Windows XP." And we've heard that before, 95%. "With the looming end-of-life for Windows XP slated for April 8, 2014" - 13 days away - "the banking industry is facing a serious risk of cyberattacks aimed at their ATM fleet." Can you have a fleet that doesn't go anywhere, Leo?

Leo: You can't use it as a collective noun. Not a fleet. A murder of ATMs, maybe.

Steve: "This risk is not hypothetical. It is already happening." Now, it is already happening. Now, it is already happening, despite the fact that we haven't yet reached April 8, so we haven't yet cut off any of the security patches, which won't start getting cut off technically until the month afterwards because after all we only get them every month now. So in reality we shouldn't really be getting anything until May. But notwithstanding that, the risk is not hypothetical. It's already happening. "Cybercriminals," says Symantec, "are targeting ATMs with increasingly sophisticated techniques." Now, that's doubtless true.

And then they explain: "In late 2013 we blogged" - very much like this one - "about new ATM malware in Mexico, which could let attackers force ATMs to spew," Leo. It's not - it doesn't come out in the little slot where you have to pick up the twenties.

Leo: No, it flies out.

Steve: You've got to get a bag, and you stand back because this is going to come spewing out of the ATM.

Leo: Wow, I can't wait.

Steve: On demand. And that's important, because you wouldn't want it to just spew if you weren't around. You'd want to wait until you demanded it to spew. And then, when you've got your cash bag ready, out comes all the money. And it uses an external keyboard. So it's like, wow, you know, you hook an external keyboard up to the ATM.

Leo: That's not going to attract attention.

Steve: And then you give it this "spew cash" command and stand back. "That threat," says Symantec, "was named Backdoor.Ploutus."

Leo: What?

Steve: I don't know, P-l-o-u-t-u-s. Backdoor.Ploutus. "Some weeks later, we discovered a new variant which showed that the malware had evolved into a modular architecture." And, after all, modular computing, that's all the rage now, so the bad guys have that, too, Leo. "The new variant was also localized into English language, suggesting" - oh, I guess you had to do "spew cash" in Spanish previously, and that confused some of the cybercriminals, so now they've translated it into English - "suggesting that the malware author has expanded their franchise" - ah, they're franchising. That's nice.

Leo: Oh, yeah.

Steve: There's a huge demand for cash spewing.

Leo: Yeah, cash spewers.

Steve: And you'd want to be able to just franchise this. You really don't have to make the burgers yourself, you just sell the opportunity. "The new variant was identified" - are you ready?

Leo: Yeah.

Steve: "Backdoor.Ploutus.B, referred to as Ploutus throughout this blog." They're going to simplify it for us.

Leo: Oh, thank you, thank you.

Steve: Yeah. So here it is: "What was interesting about this variant of Ploutus was that it allowed cybercriminals to simply send an SMS to the compromised ATM, then walk up and collect the dispensed cash. It may seem incredible..." I'm reading this literally from the blog. It may seem incredible, Leo, "...but this technique is being used in a number of places across the world at this time." So I dug deeper. You must first attach a cell phone via USB to the ATM.

Leo: Oh, well, fortunately there's a convenient USB port on every ATM machine in the country.

Steve: Then you're able to send an SMS message to the cell phone you have previously attached via USB. And, by the way, that allows it to stay charged because otherwise you've got to get your cash out in a hurry.

Leo: Oh, so it has to be a USB with power. Okay, I'll make a note of that.

Steve: It's got to be a powered USB. Find a cell phone, attach it to your ATM, and then...

Leo: Make sure, by the way, it's a burner in no way affiliated with you.

Steve: Good point, because when they wonder why the ATM has spewed itself to exhaustion...

Leo: It's spewing.

Steve: ...they'll come and find Ploutus.B in the backdoor and wonder why there's an extra cell phone attached behind the thing. Oh, my lord.

Leo: Fortunately, I just was over to the Bank of America, and they've put crazy glue in the USB ports on all the ATMs to prevent Ploutus. Thank you.

Steve: Yeah. And if you do see, like, a cell phone velcroed onto the front...

Leo: Just stand at the slot because it's going to spew.

Steve: Exactly. Get your bag ready. Someone who's not yet there may text it in advance of their arrival, and you can beat them to the spew.

Leo: Yeah. Now, if you had, I guess, if you could get behind the scenes at the ATM, or you could kind of break into the case - seems like it's a bad idea.

Steve: Leo, Leo, this was such nonsense. It's like, okay, yes. We replaced the mechanism behind the front of the ATM with a card sorter, which we modified to spit out money. And when we turned it on, money got spit out.

Leo: Amazing.

Steve: Yes, you know, that was our high school summer project. So, yeah. Oh, boy. So there is actually some good serious news, believe it or not, and that is that Google last Thursday, March 20th, announced that they were ratcheting up their march toward security to the final notch for Gmail. I think it was back in 2010 that they made Gmail the default, they made HTTPS the default for new Gmail accounts. They weren't going to upset anybody, I mean, and this is another theme we see over and over and over is nobody wants to break what's working. Even though what's working is security broken, they don't want to functionally break it. So it's tradeoff time. So, and we've talked about this as this has been going along, where Google has been sort of carefully moving forward.

What they announced last Thursday, which went in effect on last Thursday, is you can no longer not get an HTTPS connection to Gmail. It's no longer something you have to configure manually to get it. Anybody, even like from the first Gmail customer who never used HTTPS, now will find they are, like it or not. You can't otherwise get to Gmail. So I salute them. That is absolutely the right thing to do. And arguably they may have known of some edge cases where it would have broken things. So they just gave everybody time to get ready so that this change wouldn't break anything that was significant. And they've done that. And I imagine probably what they were doing is they set it so that new accounts would default that way. And they've had then years to collect incidences, like okay, now we're sort of softly encouraging everyone to connect with HTTPS. Who has a problem? And go explore those problems and fix them. And then, when you finally - when, like, nobody is ever having a problem anymore, then just lock it down, set it in stone, as I did with GRC.com, I guess a little more than a year ago.

So that's been done. And they did also confirm in the same blogspot posting, they said: "In addition, every single email message you send or receive, 100% of them, is encrypted while moving internally. This ensures that your messages are safe, not only when they move between you and Gmail's servers, but also as they move between Google's datacenters, something we made a top priority after last summer's revelations." And of course they're talking about the discovery that the NSA was tapping into their inter-datacenter fiber connections.

Leo: The upstream stuff.

Steve: So, nice move, yes. And remember that email as a protocol is still itself not often secured. So this is if you're in Starbucks, in their open WiFi. Now you've got HTTPS with good certificates and Google. So your email going to Gmail and it moving throughout the Gmail system is encrypted. But unfortunately, unless Gmail is connecting to another encrypted email recipient or source, it is not encrypted. And we know that the NSA is perched right out there. That's the public Internet, and they're sucking mightily on the flow of data across the public Internet. So people should still consider that their email is not secure unless they arrange, again, end-to-end encryption. That's the only way to do it is you encrypt it before it leaves you, and your destination recipient is able to decrypt it after they receive the encrypted blob. And that's going to end up being pretty much the

way the Internet gets reengineered over the course of the next five to 10 years, I think, is that's just going to be the way it happens.

A number of people mentioned an interesting-looking crypto introduction, a course on cryptography. It's at Crypto101.io. It began as a presentation, about, I think, it's a 50-some minute presentation. The video presentation is there on that page. But then it evolved into a course on crypto. It's a freely downloadable PDF for anybody who's interested in just sort of browsing around through these topics. They talk about crypto and modes and random number and all the things that we've talked about here, all pulled together into basically a crowd-supported course in cryptography. So it's Crypto, C-r-y-p-t-o-1-0-1 dot io, for anyone interested. I got a number of people ran across it and tweeted it, so I wanted to let everybody know.

And I said I was a little concerned about Snowden overstepping. This report was surprising and sobering. This was covered by The New York Times a few days ago, on the 23rd, so just recently, I guess over the weekend. The New York Times writes - oh, the headline was "NSA Breached Chinese Servers Seen as Security Threat." And The New York Times wrote: "The agency pried its way into the servers in Huawei's" - and I've been practicing my pronunciation, Leo.

Leo: Nice.

Steve: "...the servers in Huawei's sealed headquarters in Shenzhen, China's industrial heart, according to NSA documents provided by the former contractor Edward J. Snowden. It obtained information about the workings of the giant routers and complex digital switches that Huawei boasts connect a third of the world's population, and monitored communications of the company's top executives. One of the goals of the operation, code-named Shotgiant, was to find any links between Huawei and the People's Liberation Army, one 2010 document made clear. But the plans went further, to exploit Huawei's technology so that when the company sold equipment to other countries including both allies and nations that avoid buying American products the NSA could roam through their computer and telephone networks to conduct surveillance and, if ordered by the President, offensive cyber operations." Quoting the document, says The New York Times: "'Many of our targets communicate over Huawei-produced products,' the NSA document said. 'We want to make sure that we know how to exploit these products,' it added, to 'gain access to networks of interest' around the world."

So I don't know how this infringes on any American's Fourth Amendment rights, which is what Snowden has been claiming as justification for this. And I'm a little concerned that, I mean, I understand the way Greenwald and Snowden feel after, basically, with all of the attack that they have been subjected to. But they need to be careful about not going too far. The agreement has been made that all these documents were going to be vetted for their relevance to the specific issue of the NSA overstepping the law and the U.S. Constitution. This doesn't have any of that. So I was a little sad to see that. And I just thought, in the interest of fairness and balance, I ought to share this, too.

Leo: I think you could argue that a lot of the revelations about targeted attacks, including that massive dictionary of targeted attacks, kind of don't really fit that criterion of impinges in our Fourth Amendment rights. I think that is really kind of one issue that a lot of people have over Snowden's revelations. It's not the only one he did like that, frankly.

Steve: Yeah. And I just, you know, I've been a supporter inasmuch as, like from day one, the first moment we discussed this, I said I could never do this because I could never break my oath. The only reason I would have the information would be because I promised not to share it. So, period. But as I said a couple weeks ago, I liked that construction, when somebody was really complaining about Snowden, I said, look, would any of us want to turn the clock back a year and go back to knowing nothing? If there had been no Edward Snowden, think what we would not know. I argue what we do know is really important. And, I mean, a huge percentage of the security industry has been radicalized by this, I mean, truly is in the process of changing the way the Internet works. So I just think, in net, it's been good. But, boy, if you're going to do this, you do need to be careful. And I think things like this really do overstep that.

Leo: Yeah.

Steve: There's a new Firefox beta. Boy, the beta is for 29. We talked about 28 that just popped up, I got a balloon pop-up during the podcast, I think it was last week 28 came out. Now there's 29 in beta. It just seems like yesterday we were, like, talking about v4. It's like, wow, you know, they really did start rolling these suckers out. We don't know yet too much about 29. What we do know is that it's going to have a revamped user interface, which is the first time we will have seen this in some time. There's a name for it. I didn't write it down, and I can't remember it. It sort of looks like Australia, but it's not Australia. It's something. It's a word like that [Australis]. We'll figure out how to pronounce it when 29 actually happens. They have a name for their new UI. They were trying to get it out in 25, but they have been continuing to try, and it's finally ready.

So they're really excited, the Mozilla folks. They think this is a dramatic step forward. They're saying that this is the most attention given to the user interface of a browser ever. So lots of customizability, very easy to use. There's a tutorial that walks you through all the changes. They're really - they're upping their game. So I'm glad to see that. We really - I'm bullish about hoping that Firefox continues to survive. Chrome is a great browser, but it is the case that it's from an ad-funded company, and I like the idea of having an alternative to both Chrome and IE.

A brief SQRL update: I did talk about it a little bit yesterday when we were talking about my love of coding and why I code, essentially. I posted a PNG. I didn't have it for the podcast yesterday, Leo, but there in my show notes is a link, if you want to bring it up. And I did tweet it, and I shared it with the people in GRC's newsgroups. Anyway, SQRL, we're at 49 languages. We've actually registered 50, many of the later ones by popular request. I don't remember now whether someone asked for Korean, or it was one of the top 30, because I initially salted this, or seeded it, with the top 30. But Korean is the only one for which no one has yet signed up to do the translation. I imagine it will not be hard to find some English-Korean-speaking people who will be willing to make the translation, at which point that'll be 50 languages, which is pretty darn cool.

Leo: That's neat. That's really neat.

Steve: And we're at 297 translators, so just three shy of 300 people who have volunteered to help translate the strings when that happens. My work has been all about that. I've never needed to do an internationalized software. People have been able to put up with my English-only freeware and even SpinRite, my commercial software, for all these years. But I love the idea of being able to let people use something that I hope

they will be using a lot in their native language.

So what I was saying was that over the weekend I had a need to look up an index for a translated string in a dictionary where the indexes would be sorted, but not contiguous. If they were contiguous, then I could just index directly into the dictionary. If it was the 50th index, I could just go multiply by the size of the entry, and that would immediately take me to the 50th entry. But instead, for my convenience, I want to be able to allocate, like, numbers for strings in batches and leave space between them so that I can grow the user interface strings as the product matures. And that requires that I search a table of contents for the proper index. And that required a binary search. And I was excited when we were talking about on Triangulation, Leo, that I had just written a binary search. I mean, I've written them many times. This is the best one I've ever written.

Leo: What I didn't understand is that it's like a dozen lines of code. And that's in assembly language.

Steve: There it is. It's on the screen there.

Leo: It's succinct.

Steve: Yes, it is just - that's what gave me a thrill. It's just like, it works perfectly. I ran a bunch of tests on it to make sure that I didn't miss anything. But that's what my assembly language looks like. And it's the joy I get from coding is looking at a problem. And, see, everyone keeps telling me, well, that's been solved, Gibson. It's like, why are you wasting your time on that? It's like, yes, but I didn't solve it. And I want to. And as I explained yesterday, it's the journey that I love more than, I mean, I love finishing. And there's certainly pride of authorship, and I love producing something beautiful. But for me it matters what's inside.

So anyway, this is part of - this is where I am working now on SQRL. And, in fact, it's done. I now have all of the strings outside of the application, and it is instantly finding them and bringing them in. So there's a language file separate from the executable. And the Crowdin.net website will produce, out of the efforts of all these translators, will produce all of the translated strings, which my code that I have written will essentially compile into this attachment to the executable and instantly create 50 versions of SQRL, each in its own specific language.

And by the way, as I mentioned before, these translations are all public. Every single one of these strings is available to anyone who wants them. I've specifically made them public so that other people implementing SQRL clients on other platforms, which we certainly want and need, can, if they can arrange to reuse the same strings I have used in my user interface, they get all the translations for free for their own clients. So I think that that will tend to pull things together and unify the users' experience as they move from, for example, a smartphone platform over to Windows and maybe Linux and Mac and so forth.

And I haven't really shared a fun SpinRite testimonial for quite a while. I've just been sharing really short tweets because the podcasts have been so long, I haven't wanted to make them any longer. But I got a real - this is not a long story, but a nice story from a Dick Snicket in New York, whose subject was "SpinRite Saves Music." This was dated the 16th of March, so a little over a week ago. He said, "Hi, Steve. I'm a music major in

college and have a LOT [all caps] of Sibelius" - I think I'm missing a syllable. S-i-b...

Leo: Sibelius?

Steve: Sibelius, S-i-b...

Leo: Oh, it's a music app; right?

Steve: Yes.

Leo: Yeah, Sibelius, yeah.

Steve: Sibelius. "I have a LOT of Sibelius music files on my computer." Then he says, in parens, "(Sibelius is a program for music notation and scores.) My computer died a few days ago, and I had a backup on an external drive that was a week old. But during that week I had made quite a number of changes to woodwind and percussion parts of two movements of a marching band show for my former high school. The changes were quite precise, and I had sat down and consulted a professor to help me adjust the parts to the students' skill level." Which I think is really a cool idea, that you could, like, adjust the music to the people who are going to be playing it.

And he said, "In short, if I lost these changes, it would have been a nightmare to reimplement them. So I bought a copy of SpinRite, which I had heard about on the podcast. It took three hours to finish; and then the computer booted successfully, and I got all of my files back. I have since set up a backup so that all of these important files get backed up. Next step: Choreograph the drill, peoples' positions on the field. Thanks for such an awesome product and podcast. I can now get back to real work without worrying about damage to anything I do in the future. SpinRite is truly magic."

Leo: Yay.

Steve: So, Dick, thank you very much for sharing.

Leo: All right. Let's talk iOS Security, Steve Gibson, Part 3.

Steve: Part 3. So if anyone is listening to this and missed the last two podcasts, you need to go back. I'm not going to drag us all through where we've been. I'll just say that, from reading the latest version of Apple's iOS Security document, which was lengthy and full of really useful architectural details, I've developed a very, I think, complete and mature understanding of how focused Apple has been on the security of the iOS platform; that, without exception, they have shown a respect for, I mean, a technically enforced respect, with the architecture and the design, for the rights of the user. Nowhere are they receiving information that they don't need in order to deliver the services that they're offering. And this little phone that you hold in your hand is so easy to underappreciate because it is a little crypto miracle. I mean, it is, from the beginning

of its boot, all the way through, it's employing absolutely state-of-the-art cryptography in a way that shows evolution.

I mean, we need to remember that Apple has been understanding the problems and getting in front of them as quickly as they can. But the first iPhone was a closed platform. There wasn't iTunes with the App Store. Well, there's iTunes for music, but not the App Store. That really happened because people demanded that they have extensibility in this platform. And so Apple created that in a way that, as is necessary, allows them to maintain complete control over the platform. If there was any place, and all of our experience in looking at the nature of security breaches shows us that the perfect analogy is a chain of links, and that the end-to-end chain is only as strong as the weakest one. You keep pulling it, and you pull it, and you pull it, and sooner or later the weakest link is going to break. So if at any point Apple had dropped their guard, bad guys would be climbing into this environment.

And the fact is, with very few exceptions, that just doesn't happen because Apple has really raised the bar. And as I was talking about, for example, Address Space Layout Randomization, ASLR, in the context of Windows, Windows still is carrying the legacy of its past that prevents it from forcing Address Space Layout Randomization on all applications running in the OS. They would love to because then they would be more secure. If something got a foothold, there'd be less it could do with it. Apple, having the advantage of coming along later, was able to say, oh, we're going to have that from the beginning. So no developers were ever able to develop with the assumption that things were in fixed locations in memory. No one ever should have, really. But the nature is that people do. So Microsoft got caught by this. Apple has had the advantage of not being.

So there are essentially three things that I have left to discuss. AirDrop is a feature which has newly been added to the platform. And, once again, I am very impressed with the design of this. Essentially, AirDrop is an ad hoc WiFi network which bridges between devices. So it doesn't use a central access point or router somewhere. It is device-to-device, so an ad hoc point-to-point network, which is bootstrapped by Bluetooth. So it's through Bluetooth that the devices find each other and exchange their initial negotiation; agree about who they are and what they want to do; get people to agree at both ends that they want to establish this AirDrop connection. And then that negotiation provides the keying to the WiFi connection, which the devices then both bring up and find each other, establish a WiFi link, and then use that for the bulk data transfer. So, you know, beautifully designed.

To give a little more depth to this, they said: "iOS devices that support AirDrop use Bluetooth Low Energy and Apple-created peer-to-peer WiFi technology to send files and information to nearby devices. When a user enables AirDrop, a 2048-bit RSA identity is stored on the device. Additionally, an AirDrop identity hash is created based on the email addresses and phone numbers associated with the user's Apple ID." So this they're doing in order to create a fingerprint of the user without revealing the email addresses and phone numbers. Essentially this is an alias of those things.

"When a user chooses AirDrop as the method for sharing an item, the device emits an AirDrop signal over Bluetooth Low Energy. Other devices that are awake, in close proximity, and have AirDrop turned on detect the signal and respond with a shortened version of their owner's identity hash." So this is another clever thing. There's sort of an interlock step that we go through. So they respond with - so anyone who is within your vicinity will essentially send a shortened version of their owner's identity hash to you.

"By default," says Apple, "AirDrop is set to share with contacts only. Users can also choose if they want to be able to use AirDrop to share with everyone or turn off the

feature entirely. But in contacts-only mode, the received identity hashes are compared with hashes of people in the initiator's contacts." Which is perfect, and it's beautiful. So you send out a hash of your identity. Everyone who receives that sends back essentially a fragment of a hash of theirs. You then, the initiator, look through your contacts for any - and basically you make the same short hash that they made from your contacts and do a hash compare to see whether you have in your contacts any people within range that have just identified themselves with theirs, essentially.

Apple then says: "If a match is found, the sending device creates a peer-to-peer WiFi network and advertises an AirDrop connection over the Bonjour protocol. Using this connection, the receiving devices send their full identity hashes to the initiator. If the full hash still matches contacts, the recipient's first name and photo, if present in contacts, are displayed in the AirDrop sharing sheet." So you then see the recognizable identity from your own contacts list of the people whose devices have responded to your query.

"When using AirDrop, the sending user selects who they wish to share with. The sending device initiates an encrypted TLS connection with the receiving device, which exchanges their iCloud identity certificates. The identity in the certificates is verified against each user's contacts. Then the receiving user is asked to accept the incoming transfer from the identified person or device. If multiple recipients have been selected, this process is repeated for each destination."

So, again, beautifully put together. We have iCloud essentially serving the role as certificate authority. That is, remember that an SSL connection, TLS, is only as secure as the authentication of the endpoints because it's inherently subject to man-in-the-middle attacks, but the man in the middle cannot impersonate the endpoint because they have a signed certificate. In this case, Apple, through iCloud services, links their identities to iCloud certificates to solve the authentication problem. So this works. They've nailed what, again, to the user looks like a simple thing. You say, oh, AirDrop. I want to send this to someone sitting next to me. So you turn it on. They turn theirs on. They show up in your little sheet. You say "Send this to them," and they receive it. What went on behind the scenes is what I just described, a state-of-the-art, really perfectly designed security protocol. So again, hats off.

Now, that's the end of all the good news. We've reached the end of the good news. Two problems: They then discuss iMessage. Apple says: "Apple iMessage is a messaging service for iOS devices and Mac computers. iMessage supports text and attachments such as photos, contacts, and locations. Messages appear on all of a user's registered devices so that a conversation can be continued from any of the user's devices. iMessage makes extensive use of the Apple Push Notification service, APNs. Apple does not log messages or attachments, and their contents are protected by end-to-end encryption so no one but the sender and receiver can access them. Apple cannot decrypt the data." And none of that is true. Which is unfortunate. I mean, now, I understand that to some - I guess I don't understand. Other parts of this document are sufficiently detailed technically, like what I just described about AirDrop. That's a beautiful protocol description. Unfortunately, what they've just said here is not true.

Going on, they said: "When a user turns on iMessage, the device generates two pairs of keys for use with the service, an RSA 1280-bit key for encryption and an ECDSA 256-bit key" - that's Elliptic Curve Digital Signature Algorithm - "key for signing. For each key pair, the private keys are saved in the device's Keychain, and the public keys are sent to Apple's directory service, IDS, where they are associated with the user's phone number or email address, along with the user's APNs" - that's that push notification service - "address."

So then Apple says: "How iMessage sends and receives messages. Users start a new iMessage conversation by entering an address or name." That is, you know, who you want to send this to. "If they enter a phone number or email address, the device contacts the IDS" - that's that directory service - "to retrieve the public keys and APN addresses" - the push notification addresses - "for all of the devices associated with the addressee." Okay? So you're sending this over iMessage to somebody who's got an iPad, two iPads and an iPhone. All of those are listed for that person by phone number and email address and associated with them. So Apple sends you their public keys, which is cool.

So it says: "If the user enters a name, the device first utilizes the user's contacts to gather the phone numbers and email addresses associated with that name, then gets the public keys and APN [push notification] addresses from the directory service. The user's outgoing message" - that is, the message we're sending out to this person - "is individually encrypted using 128-bit keyed AES in counter mode" - which is fine - "for each of the recipient's devices." So remember, since each device has its own public key pair, the private key it keeps, and the public key it sends to Apple. Then, in order to send simultaneously to three devices, you need to individually - probably they generate the symmetric key randomly, and then they're encrypting that three times using each recipient's device public key, then bundling that all together and sending it off.

Leo: Tangling it.

Steve: Tangling it. It's then dispatched to the APN, the push notification, for delivery. It says: "Metadata, such as the timestamp and routing information, is not encrypted. Communication with the push notification system is encrypted using TLS." So what's the problem? Beautiful architecture. Very straightforward. Everyone generates a public key pair. They never share the private key. I'm sure it's all enclaved and well encrypted and all of that. The problem is, everybody shares their public key with Apple. And again, that's not a problem either, except that we ask Apple for their public key.

And that's the problem. We don't get it from them. We get it from Apple. And so it is absolutely not true that Apple cannot decrypt the data. All Apple has to do because, remember, users have no visibility into any of this. We don't see people's public keys. We're not saying, hey, does your public key start with EADFC9? This is all happening magically. All Apple has to do is give us one of their public keys, give us a public key for which they have the private key, and now they can decrypt everything. So instead of, in this example I just painted, instead of giving us three public keys, and we encrypt the message, we encrypt the key which encrypted the message three times, they send us four public keys, one of them for which they have the private key. Now they can decrypt everything, and we have no way of controlling it or knowing it.

So again, my only complaint is that they have stated something which is flatly not true in the iOS Security document. They have the capability of decrypting everything because they control the directory of public keys, and this process is completely opaque to end users.

Leo: Do you think this compromises that document in the sense that, well, if they didn't tell the truth about that, there might be other parts of the document that are inaccurate?

Steve: I don't think so because they told the truth about the architecture. They just

didn't draw the conclusion that an adversary would draw.

Leo: Right, right.

Steve: And so, but unfortunately, everyone knows that, in order to check security, you take an adversarial posture. And so in fact, if the NSA or the FBI or a three-letter initial organization compelled Apple to share iMessage flow from an individual, Apple can. So the problem is this was - and again, it's only this statement that I have a problem with. The architecture seems fine. But users, certainly listeners of the podcast, now understand that there is a tradeoff for the convenience of iMessage. And that is you do not actually have authentication. Without authentication, you do not have end-to-end security. This is what Threema gives us, is that Threema is not so easy to use. You have to arrange to share and authenticate your public keys yourself. Apple makes this transparent. Therein lies the weakness of it as a messaging system a security-conscious person can trust. Convenient, yes. Fine for everybody, yes. Secure, no. So...

Leo: Just, okay, a program note. We've got about 20 minutes. The Facebook folks have announced that they just acquired the Oculus Rift company for $2 billion plus, and we're going to do, at 3:15 your time, we're going to break in with our news department and get some live coverage of Facebook's announcement. So just a note.

Steve: Perfect, yes. And that's perfect timing for me. So I said what I wanted to say. I wanted to - the architecture is nice. The weakness is that we're trusting them with the authentication side. That's a benefit for ease of use. It's a complete collapse of iMessage as a secure messaging platform. To get that, you simply have to go out of Apple. You need to use Threema or TextSecure. And I'm still liking Threema better. It's, again, it's a little more obligation, but it's very clear, and it's now been subject to two independent security audits. I haven't talked about that yet, but I've got two security audits, and this thing just comes up five stars out of five across the board.

Leo: And I am liking TextSecure. Really, it seems like a good solution. And we know Marlie, Marlin, what is his name, Marlin Myxie Spot?

Steve: Marlin Moxie.

Leo: Marlinspike.

Steve: Moxie Marlinspike. Moxie Marlinspike. And I think he is, he's a deep sea fisher. I think that's where that all came from.

Leo: Oh, is he? Oh, interesting. Ah.

Steve: We've seen pictures of him on a boat with lines running over things...

[Talking simultaneously]

Leo: That's all right.

Steve: Yeah. I think that's where he got that. I did want to mention before I talk about iCloud, Siri, briefly, only because, in order to offer the services that Siri offers, Siri - to assume that she's a person for a moment - Siri needs to know a lot about you. If you say "Call Mary," how does Siri know who Mary is? Well, obviously there's a Mary in your contacts. Well, Siri doesn't live in your phone. Siri lives in the cloud. Which means your contacts live in the cloud, if you're going to use Siri.

And so this is, again, this is just - I'm bringing this up for the sake of completeness. Apple explains very clearly: "In order to facilitate Siri's features, some of the user's information from the device is sent to the server. This includes information about their music library - song titles, artists, and playlists - the names of reminder lists, and names and relationships that are defined in Contacts. All communication with the server is over HTTPS. When a Siri session is initiated, the user's first and last name from Contacts, along with a rough geographic location" - again, because Siri needs to know where you are if you say "Find pizza nearby" - "is sent to the server. This is so Siri can respond with the name or answer questions that only need an approximate location, such as those about the weather. If a more precise location is necessary, perhaps to determine the location of nearby movie theaters, for example, the server asks the device to provide a more exact location."

And again, this is Apple showing as much respect for the users' privacy as possible. I mean, I'm not arguing that Apple's collecting anything they don't need. But this is informative, I think. And again, not only of Apple's excruciatingly careful policy. I mean, they could ask for your exact position right off the bat. But they're saying, no, we're not going to ask unless you're asking a question that requires Siri to know more closely where you are. So that's cool. I appreciate that this is the approach they've taken.

And then they do take credit for it, saying: "This is an example of how, by default, information is sent to the server only when it's strictly necessary in order to process the user's request. In any event, session information is discarded after 10 minutes of inactivity. The recording of the user's spoken words is sent to Apple's voice recognition server. If the task involves dictation only, the recognized text is sent back to the device. Otherwise, Siri analyzes the text and, if necessary, combines it with information from the profile associated with the device. For example, if the request is 'Send a message to my mom,' the relationships and names that were uploaded from Contacts are utilized. The command for the identified action is then sent back to the device to be carried out."

So I thought it was - it's interesting that, when you think about it, it's easy to take it for granted without realizing, wow, "Send flowers to my mom," that requires a lot of contextual information that Apple obviously has to have. And that's not being done in the phone. It's leaving the phone. But again, Apple is minimizing what they take and discarding it a short time after they've used it. So, good to know; but, again, evidence that, where they can, they really are doing the right thing.

Now, lastly, this is enough of a concern that it would, I think, lead people to consider whether they want to change their behavior, certainly based on what their behavior is. And that is the one aspect of iCloud architecture which is the Keychain syncing. Unfortunately, super handy. I mean, really, if you're an Apple fan person, and you've got lots of iOS devices, you've got them all synced into the cloud and bookmarked. I mean,

I'm using this stuff more and more. I'll, like, deliberately bring up a page on an iPad I always have next to me here where I'm working, drop it onto my reading list, knowing that the iPad that lives in the car, when I go out for a meal, will have that link, and I could pick it up and read the research that I was deferring till then. So I love all this stuff.

But the Keychain is a concern in a pure RSA sort of worry mode. Everywhere that we have encountered, they have been using the right crypto. In every instance. And in fact, I re-read the paper, the entire thing, after I stumbled my toes over the use of the wrong elliptic curve for protecting the Keychain because it is the only place in Apple's entire architecture they use the wrong elliptic curve. And by "wrong," I mean one that came from the NSA, which no security expert now trusts.

Leo: But in that time that they used it, it was unknown; right? We didn't know that they had compromised these. Is that right, or no?

Steve: No, I'm afraid that - didn't we get all the iCloud stuff relatively recently?

Leo: Yeah, but we've learned about the RSA NIST stuff...

Steve: Oh, that, yes, that recently.

Leo: ...much more recently. So curves were trustworthy until then; right?

Steve: Yes. And, okay. So let me back up a little bit. And what's creepy is even the exact way that they're using it. They said: "When a user enables iCloud Keychain for the first time, the device establishes a circle of trust." Okay. And the members of the circle are all of the user's iOS devices. "...[E]stablishes a circle of trust which will exist among devices owned by the individual and creates a syncing identity for itself. A syncing identity consists of a private key and a public key." That's okay. "The public key of the syncing identity is put into the circle of trust, and the circle is signed twice, first by the private key of the syncing identity, then again with an asymmetric elliptical key using P-256, derived from the user's iCloud account password. Also stored with the circle are the parameters, random salt and iterations, used to create the key that is based on the user's iCloud password."

Then they said: "The signed syncing circle is placed in the user's iCloud key value storage area." They said: "It cannot be read without knowing the user's iCloud password and cannot be modified without having the private key of the syncing identity of its member." So it's interesting. And unfortunately, without real details, it's impossible to know exactly what's going on. But what's odd is, first of all, they used the proper curve, Curve25519, Dan Bernstein's bulletproof, solid elliptic curve, and said so proudly throughout this document in every single other instance. Or they used large RSA bit keys, 2048 or 4096 generally.

Here, in iCloud, for no explicable reason, they have not used the good curve. They have used the P-256 curve which nobody now trusts. We know that it came from a guy named Jerry Solinas at the NSA. I mean, we've gone back, the crypto community has really looked at this carefully. And it was generated by the NSA using an SHA-1 hash where

we've been given the seed of a series of hashes, and downstream of the series is the result on which this elliptic curve is based. And I don't remember now whether it was Bernstein or Schneier or Matt. But all three of them have said no. And one of them suggested that, if the NSA knew how to find weaknesses in ECC, and there were enough of them, then they could hide the fact that they had found a weakness by using an SHA-1 hash chain and simply running it forward until it gave them a pseudorandom number that resulted in a weak key. That allows them to say, look, we didn't choose this weak key. The SHA-1 hash chain chose it for us.

So obviously it's random. Except they could have seeded - all they had to do was try a lot of them until they found one that was weak, and then present that one. And that was exactly what they did. They said, we started with this seed, we hashed it like crazy, and look what came out the other end. So trust us. And it turns out that there are, aside from suspicion, there are many characteristics of this specific curve that make it weak. And I've got links here in the show notes if anyone wants to pursue it. There's safecurves.cr.yp.to, which is Bernstein's site. There is another site that talks about it. Schneier has written that he absolutely would not trust this curve. Now, here's what's also spooky, is that they said...

Leo: You know, Steve, I hate to say this, we're out of time.

Steve: Oh, you're right. You're right. Well, and in fact we're done, essentially.

Leo: I want to know what's also spooky.

Steve: Well, what's also spooky is that this weakness allows keys to be read, but not modified. The architecture allows that. So if there was a weakness that you wanted to put in, this gets everybody's iCloud-shared Keychains, making them readable, not modifiable.

Leo: And that's what you want.

Steve: And that's exactly what you want.

Leo: Well, we're going to have to get to jailbreaking some other time, I'm sorry to say. But you can find this episode and every episode of Security Now! at Steve's site, GRC.com, 16Kb versions of the audio plus full text transcripts. You can also find the fabulous SpinRite, the world's best hard drive recovery and maintenance utility there, and all of Steve's free stuff, including his Perfect Passwords, if you want one for your WPA2 implementation. We have audio and video at our site, TWiT.tv/sn. And of course you can watch us live every Tuesday, 1:00 p.m. Pacific, 4:00 p.m. Eastern time, 20:00 UTC on TWiT.tv. Steve, we'll talk to you next week. Questions, you think?

Steve: Yes. We're going to do Q&A for sure. And I will just pull this little bit here at the end together for the beginning of next week.

Leo: Good.

Steve: And then we'll be into Q&A. Thanks, Leo.

Leo: So your questions to GRC.com/feedback. Thanks, Steve. We'll see you next time on Security Now!.