PGP: Time for an Upgrade?

Description: This past Labor Day brought some high-profile security breaches (naked celebrity photos posted online) of still-unknown origin, and other interesting news. Once Leo and I get caught up with all of that craziness, we take a look at the (sad) state of eMail privacy and encryption. We examine the past and consider what the future might hold.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-471.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-471-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here, of course. He's going to talk about iCloud security, what we know and what we think might have happened. And we'll take a look at email encryption. Is PGP the best way to go? Or are there alternatives that are better? Stay tuned. Security Now! is next.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 471, recorded September 2, 2014: PGP: Time for an Upgrade?

Time for Security Now!, the show that brings you the latest security news. And, boy, this is going to be a good one. Steve Gibson is here. He's the Explainer in Chief from the Gibson Research Corporation, creator of SpinRite, we talk about that, but also the discoverer and writer of the very first antispyware program. He's been a long-time foe of hackers and exponent of strong security. And we've been doing this show now for - we're on our 10th year.

Steve Gibson: I have webcams surrounding me, Leo.

Leo: What?

Steve: I've got webcams all over the place.

Leo: Why?

Steve: I'll tell you about that a little bit later on the show.

Leo: Oh, good. All right. Hi, Steve. Welcome. Good to talk to you.

Steve: We have a great - a bunch of news. You were talking about it through the first half of MacBreak Weekly, this iCloud iBrute iHack with fun with naked celebrities. We have the Russians are coming. Another bad problem, actually really bad problem with consumer WiFi routers. A recent concern over fake cell phone - everyone calls them "fake cell phone towers," only because they typically are towers. In this case they're actually base stations. A CryptoLocker clone. I wanted to chat with you briefly about something that I forgot to pick up last week, which is China's operating system. And I've heard you talk about it several times on other podcasts.

A brief update on SQRL, which is why I'm surrounded by webcams. New trouble with RAID 5, which is interesting. And then our main topic is to look at email encryption. It's sort of nominally about PGP because it was triggered by Matthew Green, who's the cryptographer at Johns Hopkins. It was his blog posting, which itself was triggered by the Google and Yahoo! announcements that are going to be offering email encryption. So we're going to sort of do a retrospective and prospective look at that whole issue of email encryption. It turned out to be a little bit less specifically about PGP. PGP is just sort of a convenient whipping horse. Is that a phrase? Whipping?

Leo: Yeah, whipping - whipping post. Whipping - stalking horse or whipping post.

Steve: Something, I don't - stalking horse.

Leo: Yeah, I don't know.

Steve: To mix some metaphors. So, yeah, lots to talk about. And I think a great podcast.

Leo: Yeah. And I have a lot of interest in email encryption. I've used PGP to sign my mail for years. And I am very well aware of all the issues, just from a point of view of a user, that come up. I've recently switched over to S/MIME just because it's a little more straightforward. So I'm curious about this.

Steve: Yes, you mentioned that before. And I love the way Matthew concluded his blog. He said: "I realize I sound a bit cranky about this stuff. But as they say, a PGP critic is just a PGP user who's actually used the software for a while."

Leo: Yeah. Oh, I can - I'll give you some color play-by-play as we go along here.

Steve: Yeah, from the field.

Leo: Yeah. Oy. All right, Steve. Where do we start with this big lineup here?

Steve: So you were talking for the first half of MacBreak Weekly pretty much about this iCloud breach. And I think it's significant that we don't really have proof yet from anything that Apple has told us that, like, the nature of the breach. Certainly it was wrong for the Find My iPhone interface not to have any sort of lockout. And so what we do know - and this could be completely tangential to the fact that a hundred celebrities got personal and private parts uploaded to the Internet. But what we do know is that on GitHub there was a project called iBrute. And it just leveraged the fact that the API for Find My iPhone had no rate limiting and no failure lockout of any sort. Which clearly was an oversight on Apple's part.

Again, it's one of those where we're back to our classic tradeoff of security and convenience. If they have a lockout, some people are going to lock themselves out. And whereas, like, the next guess of their many passwords that they use might have been right, but the lockout occurs just before they get to that one, and now they can't get in, and then they've got to go to do "I forgot my password" at Apple. So...

Leo: And we should point out that, and we read this on MacBreak Weekly, but Apple has put out a statement in which they said there is no flaw in iCloud or Find My iPhone. They, by the way, fixed that rate limiting issue yesterday.

Steve: Yes, immediately.

Leo: And they said that it was in fact an individualized attack that utilized secret questions, among other things. So...

Steve: Well, and that - yes. And that leads me perfectly into the next point, which is we covered a couple weeks ago the news about the discovery of the massive database in South Central Russia of nominally 4.5 billion email addresses and names, which when filtered and dup eliminated and sorted through, still left us with 1.2 billion names. Now, the reason I bring this up is that there is a domain name registrar, DNS, and website hosting provider by the name of Namecheap who yesterday blogged their urgent security warning about the fact that their site was under attack. Mark Russell, who's the VP of their hosting services, posted yesterday, on Labor Day. And what he put down was just very responsible, and I think so clear, I just want to share it.

He said: "Overnight, our intrusion detection systems alerted us to a much higher than normal load against our login systems. Upon investigation, we determined that the username and password data gathered from third-party sites, likely the data identified by The Register, is being used to try and gain access to Namecheap.com accounts.

"The group behind this is using the stored usernames and passwords to simulate a web browser login through fake browser software. This software simulates the actual login process a user would use if they are using Firefox/Safari/Chrome" - any of the standard browsers - "to access their Namecheap account. The hackers are going through their username/password list and trying each and every one to try to get into Namecheap user accounts.

"The vast majority of these login attempts have been unsuccessful as the data is incorrect or old, and passwords have been changed. As a precaution, we are aggressively blocking the IP addresses that appear to be logging in with the stolen password data." Because those can't be spoofed. You have to have a TCP connection, so you have the IP address of the source. I'm just adding there.

And he goes on: "We're also logging these IP addresses and will be exporting blocking rules across our network to completely eliminate access to any Namecheap system or service, as well as making this data available to law enforcement. While the vast majority of these logins are unsuccessful, some have been successful. To combat this, we've temporarily secured the Namecheap accounts that have been affected and are currently contacting customers involved, requesting they improve the security of these accounts."

So then I skipped a bunch, and then he says, he concludes: "I must reiterate" - as Apple did - "this is not a security breach at Namecheap, nor a hack against us. The hackers are using usernames and passwords that have been obtained from other sources. These have not been obtained from Namecheap. But these usernames and passwords that the hackers now have are being used to try to log into Namecheap accounts. Our early investigation shows that those users who use the same password for their Namecheap account that are used on other websites are the ones who are vulnerable."

Leo: Yeah. So it's not even a Namecheap attack.

Steve: Correct.

Leo: It's using the same password.

Steve: Well, and so what's interesting is the coincidence of these two unrelated, except in time, attacks. That is, this sounds like exactly the attack that went after Find My iPhone, which was not being monitored and was not rate limited and didn't have any X number of mistakes against the account. So, again, hypothesizing here, but this could also be - the same attack on Apple could be what Namecheap saw and is now blocking, which from their information they believe to be the use of this massive database that we've talked about weeks ago. And it was, remember, it was some guy with a security firm that was charging $120 a year to let you know. But it turns out that individuals I guess were able to check, but corporations had to pay in order to make sure that their information wasn't vulnerable. And everybody felt a little creepy about it because you had to give them all of your information, like put it in your web browser.

Leo: Put in your passwords. You've got to give them your passwords. Which is crazy.

Steve: Yeah, exactly.

Leo: So, I mean, there's reason to believe that's not - that that isn't the source of attack. It seems to be that this attack has been long going on, that - Jonathan Zdziarski's been tweeting a lot about this, and he believes that it was the iCloud backups from the victims' phones that were compromised, partly because of the kind

of data, the names of the data. Apple said it was a very targeted attack on usernames, passwords, and security questions.

So it feels to me like this might have been - I bet you they were taking advantage of the lack of rate limiting on the Find My iPhone and just brute-forcing passwords for particular people. If you say, look, I want to get Leo Laporte's iCloud backup, let's just guess, you need to guess or know the email that they use.

Steve: And yours is a big secret, Leo.

Leo: Yeah, mine's secret. One of the things Rene Ritchie said on MacBreak Weekly is that's a good reason to use an email that is not your first and last name at Gmail.com because that's the first thing people guess. And then you just keep hitting that, maybe with several people, maybe many people, just keep hitting that. And if you know who you want to get...

Steve: And when you have, for a smaller organization, for example, such as Namecheap.com, an attack on them is going to make a big blip on their radar.

Leo: They're going to see it, yeah.

Steve: Yes. But with an organization as large as Apple, with the ecosystem as large as theirs, it's going to be, as long as it isn't like a denial-of-service attack of guessing usernames and passwords, if it's just a trickle, and over time they crack them, then that's going to slip under the radar. So, I mean, the really good thing that came from this is that something that should have had an account lockout now has one. Though, again, it's a tradeoff. Users are going to be inconvenienced. But you could argue that, well, yes, they deserve it if they are unable to log into their account after however many attempts Apple has set. Do we know how many that is? Did Jonathan post?

Leo: On the new standard? I don't know. No. Apple says to protect against this type of attack we advise all - and I think this is a clue - all users to always use a strong password and enable two-step verification. It'd be my guess - you know, if you have a strong password, a brute-force is going to be very difficult. So it'd be my guess that Apple's in a way hinting these people didn't have strong passwords and were not using two-step verification. Those things can be turned on. That's just a matter of changing your password and making it stronger.

That's what Zdziarski posted on his tweet is that, A, people should not only do that, lock down your account, but to also delete your iCloud backups of your iPhone and iPad because that is apparently what people are grabbing. And it should be pointed out that they got - if they got that, they got a lot more than just photos.

Steve: Yes. Yeah. The photos were just what was juicy that they were able to share. It's interesting, too, that this happened all at once. I mean, I wonder if this was a Labor Day release of data that was accumulated over a much longer period of time.

Leo: That's what I think, yeah.

Steve: Yeah.

Leo: It's hard to tell because it was a backup. So a lot of the contents were old, some as many as three years old. But that's a backup; right? You know, if you kept it on your phone for three years, and you backed up your phone, it's going to be there. So I don't think...

Steve: It was also noteworthy that apparently some of the imagery showed people with non-Apple phones. And so, I mean, we don't absolutely know where all of this came from. We just know that iCloud seems to have been implicated.

Leo: Well, Zdziarski says that the filenames, he has at least one complete dump, and that filename indicates an iCloud backup.

Steve: Ah, okay.

Leo: And nobody's talked about anything else except iCloud. I haven't looked at the pictures, so I don't know that there are non-Apple phones in them. But, yeah. It feels - and by the way, from the information Zdziarski is tweeting out, I think they're going to catch the person who did this. Lots of information was left behind in this dump.

Steve: Good.

Leo: For instance, he knows what photo editor was used on some of these photos. There's a UUID in these files. There's a lot of information that law enforcement can use to get this person.

Steve: Really does sound like this had been pending for a while; that, like, this was all collected over time and then dumped out at once in order to make quite a big splash, as I guess it did. And it was posted to 4span?

Leo: 4chan.

Steve: 4chan, right.

Leo: And there's nothing you can do much about 4chan /p/. That's kind of a Wild West there.

Steve: Out of our control, yeah.

Leo: And I don't think - I don't know if 4chan's taken it down. Every other source has taken it down. And even people like Perez Hilton, who initially posted links, have taken their links down, realizing, you know, he says, "I am sorry, I should never have posted that." I think there's a general attitude that this is a bad thing, and nobody wants to participate in this at this point.

Steve: 4chan was charging $95 for access to them.

Leo: Were they?

Steve: So, yeah, they were making...

Leo: I don't think that's the case. No, Steve, I don't...

Steve: Trying to monetize them.

Leo: 4chan doesn't do that. I don't think it was 4chan. I think it may be that the person who put this up offered it for sale. And we know that they offered it to TMZ for hundreds of thousands of dollars. I don't think 4chan does that.

Steve: I ran across that this morning in my research.

Leo: Yeah, I don't believe that's the case. I think the person on 4chan may have asked for money.

Steve: Ah, okay.

Leo: 4chan is not - is a wide-open forum. It's not - and I would be very surprised if Moot would do that. That does not seem like the kind of thing they would do.

Steve: So we do have a new, very, very worrisome WiFi exploit against consumer routers. And this is another problem with WPS that we've spoken about before. In fact, early in 2012, our January 10th podcast, No. 335, the title of that was "WiFi Protected (In)Security." And our long-time listeners may remember that there was a very clever hack that was found, and some software called "Reaver" was created. We joked at the time that apparently the guy who did it, who generated the hack, was a fan of "Firefly," since the Reavers were some of the creepy bad people on the "Firefly" sci-fi series.

And so you'll remember that the secret here was that the WPS key is eight digits. It's an eight-digit code printed on the labels of many WPS-enabled wireless routers. And the idea is that, if you provide this PIN, this eight-digit PIN to the router, and so you're

proving that you're physically present and able to read the label on the router, you provide the PIN to the router through your WiFi client, like running on Windows or something that has a WPS option, and the router will then provide your client with its passphrase, which is like - so the whole idea is it's a simple way to prove you have control of the router, and it just hands you the passphrase, which allows you and encourages you to have a crazy passphrase that you're unable to enter conveniently otherwise.

Okay. So the problem then was that, first of all, the eighth digit of the eight digits wasn't actually information-bearing. It was a checksum on the other seven. So we can kind of forget about that. So that left us with seven digits, and there are 10 million combinations of seven digits for brute-forcing. And if you made a mistake, if you typed in the wrong one, then the router would lock you out for I think it was like 43 seconds for some reason. I mean, it's like some chunk of time, so that there was a penalty, but not onerous, for entering the wrong one. So that the typical user would be told to wait a minute. They'd wait, then they would fix their typo and hopefully put it in correctly the second time. The point is that that lockout was a long enough delay that, when coupled with the 10 million combinations of seven digits, where the eighth one can always be calculated as a checksum, prevented anyone from getting access to your passphrase.

What was discovered back at the beginning of 2012 was a glitch in the protocol. And I'm sure, as I went back, as I refreshed my memory, it's like, oh, yeah, I remember this. There was a multiphase handshake back and forth. And after the first four digits were put in, it was possible to determine whether those were right, separate from the entire seven. And that was the fault. That was the flaw because what that meant was that you could crack this in halves. There were only 10,000 combinations of four digits. So that's way fewer than 10 million. So you could brute-force over some period of time. Average is going to be half of that, assuming an average PIN. So in an average of half that many guesses, so 5,000 guesses, times the minimum lockout period, allowed you to get the first four. Then you only had the last three. And so that was only a thousand of those.

So overall, in 11,000 guesses times the lockout period, a person could get in. And it worked perfectly for everyone who had WPS enabled. And at the time we said WPS was not a good protocol, clearly a worrisome tradeoff between usability and security. Everybody should turn it off. That was our advice then.

Now what has come to light is a new problem. And Dominique Bongard, who's the founder of Oxcite Security in Switzerland, has made a presentation about this. So this is now in the public. And essentially he has found for some routers he can crack WPS in one second, basically just no time. Now, there are two classes of routers where he has found a vulnerability. The problem is that routers based on a Broadcom chip - and apparently we're sort of at the first stage of this. Broadcom has been notified.

The problem with the firmware, and in fact this was the same problem we talked about two and a half years ago, and that is they implemented the demonstration firmware. It was just sort of - it was the proof of concept. It was like, here's some sample firmware for how you would implement. I remember that this came from Intel. And it was like the WPS firmware. And it was like, do not use this. But here's some sample firmware. But we're not doing error checking and blah blah blah because we don't want to bog the code down with a lot of extra stuff you actually need to have because then it'll be less clear what's really going on. So this was just the demo firmware that got written into the EPROMs of these routers.

Well, we sort of have the same thing because it turns out that the pseudorandom number generator in the firmware of this class of routers that is based on the Broadcom

chip is just a 32-bit simple linear congruential PRNG. And I've talked about the trouble with linear congruential PRNGs. That's one where you have a value, and you multiply it by a constant and then add another constant to get the next value. You discard the overflow from that. And, I mean, it's not cryptographically secure in any way. It produces sort of a - it produces an unpredictable number if you don't know the constants, unless you look at a few of them, and then any cryptographer can reverse-engineer the constants. So, I mean, even if the constants varied, but they don't. They're burned into the code.

So this thing produces a fixed pattern of, quote, "pseudorandom numbers," I mean, very pseudo. And it turns out that the nonce which is generated by the access point at the beginning of the WPS negotiation is taken from the pseudorandom number generator, and then two other crucial nonces for the handshake, which have to be secret, are subsequently taken.

Well, the act of taking the data from the pseudorandom number generator shows you its state. And so, and the access point's first nonce, used at the beginning of the handshake, is public. So the attack is you tell the access point you want to initiate a WPS negotiation. It hands you the nonce, which it just got from its lame pseudorandom number generator. That gives you the state of the pseudorandom number generator, and then you know the two secrets which it's about to obtain subsequently from the pseudorandom number generator. So it's possible in less than a second to completely compute the secrets that the handshake uses and crack the PIN.

So this is the condition of a class of routers based on the Broadcom chip. The other class turns out to be even worse. And Dominique has not disclosed the manufacturer. He's giving them time to remediate the problem. That one is based on a linear feedback shift register. So it's a common technique used, again, for generating weak pseudorandom numbers. And they can be stronger than a linear congruential pseudorandom number generator. For example, cellular phones use linear feedback shift registers because they can run very fast. If designed well, they can generate very good pseudorandom numbers very fast. But if it's ever possible to know the state of the feedback shift register, then you know all of its future.

It turns out that this router initializes the shift register to zero whenever it's powered up. So cracking WiFi that's protected by this one is as simple as somehow getting it to reboot. You could, for example, briefly interrupt the power, "trip the breakers," as Dominique puts it, and then turn the power back on. The router will reboot and come up in a known state and generate absolutely repeatable pseudorandom numbers for its WPS negotiation, and again you're in.

So the takeaway is there is another bad WPS breach. We don't yet have model numbers and manufacturers. We only know Broadcom chip, and this other one is as yet unknown. But if anyone didn't turn WPS login or negotiation off in their router firmware two and a half years ago, now would be a good time to do it. So, and I did check mine to make sure mine was still off because things change over time. In fact, I think I have a different router than I did three years ago, and mine doesn't have WPS as an option. So I was glad for that.

So again, it may not affect everybody. But WPS has been a troubled protocol in the past. Its manufacturers responded by making a harder lockout and a longer lockout rather than fixing the protocol, which they actually couldn't change because it was already baked into the clients that were expecting it to be available. In some cases you have to press a button to enable it, and then it's only on for some length of time. That's certainly better than the protocol just sitting there wide open, happy to negotiate with any

stranger who comes along, which is the typical case several years ago. So, good idea just to turn this off. Or turn it on during the brief window of use where you're negotiating a WPS-capable client to your router and then turn it off again because it's just a bad thing to leave on.

Also in the news was several different reports of so-called "Stingray," which is the term they're being given, fake cell phone towers. Again, they're not towers, they're typically just base stations. But it came to people's attention that - and there were some headlines that were a little bit overwrought, again, talking about how they could be used for installing malware in Android phones, which goes way too far. It does speak to the concern that we've talked about on this podcast before of the so-called "baseband radio."

All cell phones, smartphones, have an operating OS that we talk about, Android and iOS and so forth, and another sort of a lower level real-time OS that runs the cell phone circuitry, the cell phone radio. And, I mean, it's intelligent. It doesn't get nearly as much scrutiny as it should. It's just sort of clouded in secrecy. And so the concern is that there could be vulnerabilities there that haven't had the attention that the OSes that we're used to interacting with and that we run apps on have had. And that, if you had malicious cell phone base stations or towers, that is, things mimicking real cell phone towers, then they could get up to some mischief.

So it turns out that there actually is an infrastructure of these. The FBI; DEA; the U.S. Secret Service; Immigration and Customs Enforcement; the U.S. Marshals Service; the Bureau of Alcohol, Tobacco, Firearms and Explosives; the Army; the Navy; the Marines; the National Guard; U.S. Special Ops; and, not to be left out, the NSA, all are known to have these things. On one website, there's a site called Meganet.com that sells stuff into law enforcement and government. And they have something called the "VME Dominator," is their name for this.

And they said it's a "real-time GSM A5.1 cell phone interceptor. It cannot be detected. It allows interception of voice and text. It allows voice manipulation, up or down channel blocking, text intercept and modification, calling and sending text on behalf of the user, and directional finding of a user during random monitoring of calls."

The ACLU is all up in arms over this, feeling that it breaches U.S. citizen sovereignty. They have a map showing the 18 states in the U.S. that are known to have government facilities using this. And so apparently this is something that law enforcement uses for tracking people. Essentially they're mobile fake cell towers that users' cell phones will associate with, believing that they are legitimate. And then essentially someone who is not the cellular carrier has decrypted access to the otherwise encrypted over-the-air data, which they're able to do anything with they want. So, you know, for IME tracking. And I guess with a couple of these you're able to pretty much zero in on the location of someone. So you can imagine that the government is using this technology and taking advantage of the fact that so many people are carrying cell phones with them all the time.

There is a new CryptoLocker clone. The site that we like a lot, really great work by Lawrence Abrams, that's the BleepingComputer site. And thanks to Simon Zerafa, who follows the show and tweets all of his finding to me, for pointing this out. There's ransomware which is absolutely not CryptoLocker, but it's now taking advantage of the reputation of CryptoLocker to call itself CryptoLocker.

Leo: That's funny. Now, that's funny.

Steve: So it works nothing the same. I mean, like all of the - it's completely rewritten, clearly from other people, uses a different network. It actually leverages Tor for its stuff. It puts a .encrypted extension on the encrypted files, which CryptoLocker doesn't and never did do. They're charging 1.8 bitcoin, which at the moment is...

Leo: Oh, that's a lot.

Steve: Yeah, about $864 U.S. And it calls itself, it says, "You've been infected by CryptoLocker," just because now everybody knows about that.

Leo: It's a brand, yeah.

Steve: [Laughing] It's the CryptoLocker brand.

Leo: It's still strong encryption? It's the same effectiveness?

Steve: Yep. It looks like it's bad stuff. It operates differently, though. As I said, it's very different. So there isn't the same sort of - it's not clear to me that there's the same sort of pre-encryption handshake. What we do know, and this has just been spotted in the wild, so it hasn't had deep analysis yet, but it puts a cookie, PHPSESSID cookie on your machine so that, when you then go to the site to make your payment in order to pick up a downloadable executable that will give you your files back, your browser provides this PHPSESSID - P-H-P-S-E-S-S-I-D - cookie as part of its transaction to the site, which will then look you up in its database in order to provide you with a matching decryption for your files.

And Lawrence noted in his blog posting about this, that was yesterday - boy, I tell you, Labor Day was a busy day all around - that currently it is not deleting Windows Shadow Volume copies. So it is possible to do some recovery of encrypted files if volume shadow copies are still around and containing files that this thing has encrypted. So anyway, it's looking like, as you said, Leo, CryptoLocker is becoming sort of a trademark of...

Leo: Well, we all know what it does; right? I mean...

Steve: Exactly, bad ransomware. Yeah, exactly. It's like, oh, yeah, okay. CryptoLocker, that's bad.

Leo: That's bad. We know that's bad. All right. Let's continue on. Steve Gibson with his litany of security messes.

Steve: I was going to say, my own particular use of OpenVPN is for authentication because I want, if I'm roaming, I want to be able to access GRC's internal network. Several times...

Leo: Well, that's how businesses use it, yeah.

Steve: Yeah. Several times when I've been up in Petaluma with you I've needed to tweak something on GRC's servers. And so I use the certificates, and of course passwords on top of that, to authenticate me to GRC's network remotely. And I do the same thing for home, when I just need to get to my network when I'm out roaming around.

Leo: I am sure that's how most people first experience VPN is that that's how they got into the office network securely from home.

Steve: Right, right.

Leo: And then they thought, oh, I could continue to surf, and I'd be looking like I was coming from the office. But why not, I mean, this takes it completely out of your own domain, even out of the country, which is I think a good way to do it.

Steve: Yeah. So let's you and me talk about China's doing their own operating system.

Leo: Yes.

Steve: Because you've talked about it a couple times. And I just don't understand why they wouldn't adapt a regular Linux desktop environment. I mean, like...

Leo: I think they've been doing a...

Steve: Why reinvent chopsticks?

Leo: Right. They were doing a Red OS some years ago, we knew this, based on Linux, a version of Linux.

Steve: Okay, yeah.

Leo: I wouldn't be surprised if this new OS, who knows if it's written from scratch or based on a Linux kernel or what. I think, if China's going to do an OS, they want to do it for a couple of reasons, mostly because they want to control it; right?

Steve: Yes, yes.

Leo: And so you could certainly start with a Linux code base.

Steve: Oh, god. I can't imagine not. I mean, it's open.

Leo: I mean, so much good work has been done.

Steve: Yeah, exactly. It's all there.

Leo: Well, open is not what China wants.

Steve: No, no. But, I mean...

Leo: You could start with it.

Steve: It's open, yeah, it's open in terms of it being a fully functional, beautifully designed, I mean, state-of-the-art robust platform. And open in terms of them being able to scrutinize it to make sure there's no NSA tricks in there, which they can't do with Windows, inherently.

Leo: Or any closed source OS; right?

Steve: Right, right.

Leo: And I made the point, and you probably heard it, that China has the expertise. There are so many good engineers. There's a billion people. There's so many great engineers there. And Paul Thurrott was saying why do this and that, I mean, look how long it's taken to create Windows and make it robust and reliable. Yeah, but China has a little bit more resources than Microsoft.

Steve: Oh, my god. All they have to do is stop attacking us and just work on their operating system.

Leo: I bet they can do both. There's also the issue of the language, which I mentioned to Paul, which is that the Chinese language is notoriously difficult because it's not an alphabet, it's ideographs. And that makes it very hard for everything. And it's one of the reasons they're actually very aggressively pushing English-language education in China.

Steve: Yeah. The question for me is not why are they doing it now. It's not what took them so long. I mean, with this obvious sort of adversarial tension that exists certainly in the cyber level with the West, it's like, why are you using Windows, of all things? I mean,

that just makes - that's crazy. So, yeah. It'll be interesting to see what it is. And I will note Paul's comment that, while it's true that it's taken us a long time to get Windows, I would note that we had Windows a long time ago. And all Microsoft has done since is just to mess with it in order to make us continually upgrade it and generate revenue for them. I mean, it's not clear to me that this does a lot more today than 3.1 did 20 years ago.

Leo: Right, right.

Steve: I mean, yeah, it's fancier, and it's got more I/O and memory and so forth. But I guess my point is that anyone starting today could do an OS from scratch because we know how to do that now. We really weren't sure how to do what we have today 30 years ago. There's been, you know, Microsoft is carrying a huge load of legacy forward in the same way that Intel is with an instruction set that they now regret. But both companies, Intel and Microsoft, are succeeding specifically because everything that they ever did still works. And nobody wants to give that up. So I'll be surprised if it turns out that this isn't a Linux-derived desktop environment. It's like, that's what I would do. They're not asking me. But it just seems clear.

Leo: Yeah, a lot of the problems have been solved. And let's not forget, Linux was written by one person, a graduate student. And, I mean, now of course there's a lot of other people contributing to it. But the Linux kernel itself was a single-person operation. Operating systems are pretty well understood by now.

Steve: Yeah. And in fact a microkernel, there has been some really neat work recently on a microkernel where virtually everything, essentially anything that doesn't have to be in the kernel, isn't. So it's a scheduler, and it's process management. Everything else you can do outside in so-called "user space" and create a very robust, very small kernel that computer science graduates could do as their project.

Leo: Right, right. Simple enough.

Steve: So it turns out that outputting a QR code, Leo, is way simpler than inputting one.

Leo: Oh. Tell me about it.

Steve: The reason I'm surrounded by webcams is that I decided that to make my SQRL client complete, I had to allow importing of QR codes, just, I mean, for convenience sake. The printed form of a SQRL identity that I held up to the screen last week is both in a QR code and a simplified base-64 ASCII, so someone could enter that if they want to, but the QR code is simple. But say somebody develops or designs or sets their SQRL ID on an Android phone or an iOS device first. I mean, they download SQRL, they get their ID set up, and they want to transfer it to their Windows machine.

Well, arguably there are ways to do that, through email, and Windows has a file system. And so there's probably a way to do it. The simple way is to show your identity on the screen of your phone to your webcam on your laptop. And most Windows machines now,

laptops have webcams built in. So anyway, I bit the bullet, I think it was shortly after the podcast last week. And oh, my lord, it's been an adventure. I have nearly finished. It turns out that I tried to use the earlier standard for Windows, which is called, you'll remember, Video For Windows, VFW. And it's simple to use.

Leo: Wow. I haven't heard that in a while.

Steve: I know. Well, and so I figured go to the lowest common denominator. And the first, the webcam that I...

Leo: Is it still in Windows?

Steve: Yeah, Video For Windows is still there. And again, that's what I'm talking about.

Leo: Thank you, Microsoft. You never kill anything.

Steve: Legacy, exactly. And the other thing is this all runs under Wine, so Macs can use it that have Wine, and Linux can use it. And I figured, again, lowest common denominator. If they've supported any kind of video stuff, it will be the oldest stuff, the Video For Windows library. It turns out, though, that my Video For Windows implementation came right up, and I'm looking at video from my webcam. But I've got a bunch of webcams. It didn't recognize any of the other ones. And so it's like, oh. And so I found some random, like, video capture thing that I downloaded, and it recognized all of my webcams.

So I thought, okay, I just can't - I'm not going to go halfway. So then I had - the only way to do this correctly is with something called DirectShow, which is written on top of DirectX, which is written on top of COM, which is Microsoft's Common Object Model, which is, you know, nobody likes who's ever programmed, and they've never done it in assembly language. So I've implemented all of that infrastructure now in assembler, and I now have webcams. All the webcams run, and I'm looking at video in a little window. And just last night I ended up capturing the frames as they're going by, and now I feed it into a QR code recognizer. So, whew, I'm close.

Leo: We should say this is just one implementation. People are going to be able to implement whatever they want; right? This is your reference implementation.

Steve: Absolutely. And, I mean, I'll be surprised if anyone else does it in Windows because, I mean, GRC will have one which is absolutely complete. But I imagine they'll be for Mac and for maybe other ones natively for Linux, rather than Wine implementations. And certainly Android and iOS. There's already one for Android, and I know that there are people working on them for iOS. And then a bunch of people doing the server-side stuff.

So I've wanted to mention briefly two things that occurred over on the hard drive side. One was a question I got, "Why can't SpinRite be built into a drive?" And also, "What about an OS?" And that's come up from time to time. And the reason is that a drive,

while it's very capable, is actually not very smart. And one of the things that software running on a host computer can be is way smarter than the drive is able to be. And one of the ways you get smartness is from having access to huge amounts of memory.

And when SpinRite comes to a so-called grinding halt and fires up its DynaStat system, that Dynamic Statistics system, it is taking samples of a sector which is refusing to be read and filling the system's memory with those samples and dynamically analyzing the samples in order to essentially reverse-engineer what data was most likely stored in the sector that refuses to give it up. So that requires a bunch of sort of both heavy-duty number-crunching and access to a huge sample pool that no drive could spend the money to have onboard. And it just wouldn't make sense. So really SpinRite and the drive are working together, where SpinRite is functioning sort of as the drive's brain extension in order to help it recover sectors that it's having problems with.

And, now, why not do that in the OS? And that may be something, I mean, I've always been thinking about that. The problem is SpinRite is now becoming multiplatform officially with v6.1, which I will get back to the second I finish the SQRL client, so that it runs on Windows and - or I should say a PC and Mac hardware both, and is OS agnostic. So SpinRite is still not caring about your operating system. It'll know in v7 more about the file systems. But in order to build it into the OS, it would have to be running in all of these different kernels. And I'm going to need something to do after SpinRite 7 is finished, so I think building it into the OS makes a lot of sense. So I think we'll get there.

And lastly I wanted to talk about a really interesting piece that I read about the end of RAID 5, arguing for why we need RAID 6. And I thought it made a really good point. What's happening is, as drives have been growing in size, as we know they have been, like phenomenally, the unrecoverable error rates have been creeping upwards also. So just because the bits are becoming so small in order to get these ridiculous - you probably saw that Western Digital, I think, no, I think it was Seagate, has now announced, and I think they're starting to ship, the first eight...

Leo: Eight terabytes.

Steve: ...terabytes.

Leo: It's unbelievable.

Steve: Right. It's like, oh. Okay.

Leo: As soon as I see that, I go, oh, Steve must be shivering.

Steve: So, well, the good news is 6.1 was benchmarking at, I think it was two hours per terabyte, which allowed it to do a 4TB drive overnight. So that brings us back into feasibility range. And then 8TB, of course, would be 16 hours. But still, if you're working eight hours during the day, you could start it at the end of your workday, and it would be pretty much done, 8TB, by the time you needed it again 16 hours later. So we're still okay.

But yes. Here's the problem. RAID 5 works by allocating one whole drive as parity. So

what that means is, if you have two drives, you add a third drive, which contains the XOR of the first two. And what that allows is any sector on any of those three drives can be unreadable. And it turns out you don't need that because you can XOR the data on the remaining drives in order to recover the data. So essentially you've created a parity drive. And so a three-drive array is not very efficient because you have three drives, but you've given up one completely, one third, just for the redundancy.

But you can run like a RAID 5 with six drives where one of the six is your parity drive, similarly. There the cost of the redundancy is only one sixth, rather than one third, so it's twice as efficient. The problem is, if a drive fails, then you do the famous so-called "RAID rebuild." And you may have noticed that RAID controllers have gotten smart. They don't even require you to normally build the array initially. They do a smart build where they only build as much as they need, just because building a RAID now on 4TB drives takes forever, essentially.

Here's the problem. With unrecoverable error rates having grown compared to the size of drives, if a drive fails and you must rebuild the RAID, during the rebuilding process you have zero redundancy, and an error which occurs during rebuild cannot be corrected. So the argument is being made that we now need RAID 6. And as I've mentioned before in the podcast, that's all I run. I run RAID 6 on GRC servers, here on my own desktop machine. That's my RAID configuration of choice. And what that gives you is two drives of redundancy. So you could lose one, no one would care. You could lose two, and you could still recover all your data.

And so just something to keep in mind, with drives being as inexpensive as they are, with controllers becoming more sophisticated, I would say take a look at RAID 6. If you don't mind sacrificing two drives for redundancy, if your array is large enough that two drives is a low enough percentage to give up, it's worth considering, just because drives are being very, very aggressive about the amount of data that they're now storing on their surfaces.

Leo: Is that 8TB drive just because of more platters or more data density?

Steve: That's data density.

Leo: Geez. Holy moly.

Steve: What they're doing is, well, actually I may have to say platters because the other thing that they've done is they've switched to helium inside.

Leo: Oh.

Steve: And helium is - it's reduced the friction of the platters with the environment. And I believe it also allows the heads to fly closer, and it has allowed them put more platters into the drive. But as they lowered the head flying altitude, which helium also does, that gets the head closer to the surface, which essentially gives it better focus. And so you also increase your bit density as a consequence of changing the atmosphere inside the drive. So, yeah, these things just refuse to give up.

Leo: We never thought that we'd get this kind of density. I mean, I thought - I remember by 2000 - I think I said, "By the year 2000 we'll be using some form of holographic storage because there's no way this physical spinning media could get dense enough." It's amazing.

Steve: And it's so cheap, Leo. My god, when you consider the cost per bit. Well, and of course the phenomenon is the cloud, and cloud storage. It's funny, too. I also read someone saying, and this refers back to MacBreak Weekly, someone said, "Rather than thinking in terms of the cloud, calling it 'the cloud' gives people somewhat of a false sense of security. Think in terms of storing your data on someone else's computer."

Leo: Right.

Steve: That's what you're doing.

Leo: That's what it is, yeah.

Steve: You're storing your data on someone else's computer. So instead of just sort of, I mean, "the cloud" sort of sanitizes it as just like, oh, you know, no one has it. No, somebody does. It's on somebody else's computer.

Leo: Right. That's a very good point. All right. Let's continue on. Time to talk about email encryption.

Steve: So the impetus for this was Matthew Green, who describes himself in this context as a somewhat cranky cryptographer at Johns Hopkins. He started his blog posting from - it's just "08," I don't know what day [August 13], but I think it was maybe the middle, early to middle of last month, of August, saying: "Last Thursday, Yahoo announced their plans to support end-to-end encryption using a fork of Google's end-to-end email extension. This is a big deal," says Matthew. "With providers like Google and Yahoo! onboard, email encryption is bound to get a big kick in the ass. This is something email badly needs. So, great work by Google and Yahoo!. Which is why the following complaint is going to seem awfully ungrateful. I realize this, and I couldn't feel worse about it."

So a couple things. First of all, what Google has explained is they're very proud of what they've done because they've bitten the bullet, not for the first time, but they've bitten, okay, a bullet of tackling crypto in JavaScript. And there's a famous blog, it's famous in the security community, blog posting, and I can't quite remember the title. It's something like - oh, no. It's "JavaScript considered harmful to cryptography." I just liked that characterization.

And it is the case that it is difficult to control what is regarded as an automatic language. Our listeners will remember that last week I shared my retrospective on how difficult it was for me to be so careful not to leave any sensitive data around in the SQRL clients, even local variables allocated on the stack. I would wipe them, if they had ever contained sensitive data, the moment I no longer needed them I would proactively zero them. And that's normally something that just sort of happens when you leave a subroutine is the

stack pointer is popped up over those buffers, those variables, and they sort of disappear. But they're still, the memory is still there. So it's like, not in my case. I'm going to wipe those out before I release them.

Well, in a language like JavaScript, you have almost no control of memory management. It's doing that for you. It's keeping reference counts. It's figuring out whether you're still using this anywhere; and, if not, it releases it. So it is a real exercise to create something which you can consider secure in JavaScript, that is, crypto. And that's essentially what this post was talking about. Also, you don't have strong typing. JavaScript is a weakly typed language, where it figures out the type, whether integer, Boolean, a string, or a floating value, from your usage, and also sort of freely mushes them back and forth, depending upon your need. So all of this is sort of antithetical to writing secure code.

Now, Google does note that, in the event of Chrome crashing, with this extension which is providing these crypto services written in JavaScript, your sensitive data could escape in a dump, in a crash dump. So it's like, oopsie, okay, well, that's a problem. But it's an example of what they just don't have control over because they're operating several levels removed from the machine where it's able to crash out from under them. And whatever they're doing at the time is part of the crash image. So, whoops. Still, they're very proud of the work they've done.

I've been using traditionally, like for the other secure stuff that I did, the Stanford University JavaScript library, which is a bunch of crypto primitives also written by really careful cryptographers who understood the challenge in order to get various sorts of crypto primitives finished. So we have Google and Yahoo! who have announced their intentions of offering some sort of service.

This triggered Matthew to sort of say, okay. Here's the problem. These guys are implementing PGP, OpenPGP, but still basically a framework which is old. And so can't we do something better? Does it make - he sort of asks the rhetorical question, does it make sense in this day and age, if we're going to do something, to do PGP? And there are a number of problems with doing that. In the show notes every week I try to find a fun picture. And in this week's show notes, the picture on the first page of the show notes gives you a sense of the relative key sizes of, like, old keys that we knew how to do then, compared to new keys that we know how to do now.

And the original PGP, I think it was a 3,096-bit PGP key, is just this massive blob of crypto-looking text; whereas a state-of-the-art base-58 key - base-58 is the conversion from binary into ASCII that avoids the use of the non-alphanumeric characters that are - because you normally have only 62. You've got uppercase alphabetic, lowercase alphabetic, and zero through nine. That gets you to 62. Then you need two more, which are normally plus and backslash, or I think plus and forward slash. But then those are not URL-friendly, so then there's a base-64 URL which uses hyphen and underscore to get the last two, to get up to 64. That way you can use six bits, a six-bit character set.

Well, this uses base-58. I'm trying to think who does that. I think it's one of the crypto currencies uses base-58 in order to avoid not only those two characters, but also zero and uppercase "O" and "L" and "1," I think. That brings them down to 58. But anyway, you can, with a state-of-the-art public key, it's just a tiny little string of ASCII that has even got the confusing characters removed from it, and it has the same cryptographic strength today that this massive 3,096 blob of public key did, PGP format public key did back in the day.

So, and I know that you, Leo, mentioned that you've sort of moved over to S/MIME. S/MIME has the advantage of better integration into email clients, and it's able to use the

traditional certificate authority public key infrastructure hierarchy or self-signed certs where you just provide your S/MIME cert to someone and say, we're going to communicate, here's my certificate. And you can do things like the recipient can take a hash of it, and then you just check to make sure that you've got the same signatures, and then you know that there's been no problem with somebody getting in between and intercepting you.

Leo: That's chiefly what I use both for is verification as opposed to encryption.

Steve: Correct.

Leo: Although once somebody has your key, they can do both. So that's nice.

Steve: Right. So...

Leo: I have to say that the S/MIME confuses people just as much as the PGP because it's an attachment, a .p7s attachment.

Steve: Yes. And all of this, I mean, this is - I think it's the overriding problem is that the fundamental architecture of email was plaintext. It was never meant to be encrypted. And notice that none of this solves the tangential problem of metadata. That is, PGP nor S/MIME deals with who is sending email to whom. That's still all in the public. And so even though the envelope itself may have encrypted contents, the envelope is still being carried by the traditional, out-in-the-open email transport system which doesn't have security as its prime focus. So, yes, your feeling is exactly what those people who have tried to use encryption come away with.

So Matthew makes the point that, regardless of the technology we use, managing keys, as he puts it, sucks. And there have been various things that have been tried. Recently there's Keybase.io is a new experimental startup for, like, creating a secure place to store keys. But we've had key servers before. We've talked about the Web of Trust notion, where you have key-signing parties in order to cross-sign keys, in order to sort of create your own non-centralized infrastructure of public keys. And then the use - because no one can, like, check a public key carefully, and they're so big, traditional PGP keys are so big. Then we use much shorter fingerprints. But the very fact that you need a fingerprint demonstrates that there's a sort of a fundamental problem here.

And then we've got the notion of autonomous or automatic management. We've talked about how iMessage, for example, manages keys for users. And while that's very convenient, it also creates a loss of control. There is an inherent tradeoff being made with the security of the system. We just assume that Apple is going to never slip a government eavesdropping key into the set that we receive when we send an iMessage. If they do, then somebody other than the set of recipients we intended is able to decrypt it. So really farming key management out to a third party, where it's done with maximum convenience, means it's done with some clear sacrifice of security.

And then the other thing that Matthew notes is that none of this technology has any concept of forward security. This is something we've talked about on the podcast a number of times. And it's now becoming a bullet point that we really have to have

moving forward. Traditional web server certificates, where the certificate was used to encrypt the conversation's symmetric key, we know that those are not safe. That model did not have forward secrecy such that, if someone obtained the certificate, if someone was storing, archiving the dialogue which they could not decrypt, and then later obtained the server certificate, they could go back and retroactively or retrospectively decrypt those conversations.

That's no forward security. Forward security means that a compromise of keys cannot be used to decrypt past traffic. And unfortunately, both with PGP and S/MIME, those protocols were designed back, like the original web server SSL protocol, without a sense of the importance of forward security. Today, that's a problem because we're in a world where it's entirely feasible for the NSA or whomever to be archiving vast quantities of encrypted traffic on the off chance that keys will become available later, and then they'll be able to go back and decrypt all of that past traffic.

So Matthew's point is, does it make sense today to be implementing old, although tried and true, standards? And I think he makes the point, with a series of examples, actually, that it doesn't. He asks, rhetorically, what should we be doing if repeating the past is not that? So he suggests a proper approach to key management. Some sort of centralized key management, he argues, would still be better than nothing. And he points to some work that has been done with Signal or Off The Record protocols. He argues that forward secrecy really, for anything being designed today, it needs to be baked into the protocol, and that it should be any precondition of a secure messaging system.

And for some reason, he has a picture of the Fresh Prince, which I guess was - I'm not sure when "The Fresh Prince of Bel-Air" was airing [19901996]. But he says, "Cryptography that postdates the Fresh Prince."

Leo: That seems fair.

Steve: And he says, "Enough said."

Leo: Yeah.

Steve: And the point being that we are using really old crypto, and we know how to do much better crypto now than we did. And he finally says, "Screw backwards compatibility."

Leo: Yay, yeah.

Steve: "Securing both encrypted and unencrypted email is too hard."

Leo: Yes.

Steve: "We need dedicated networks that handle this from the start." And I think it was, again, something that I picked up on one of the other podcasts you were doing, Leo, where it was observed that today's user is very fickle. We are very - there's a very low

friction of adoption of some other service if it provides benefits. That is to say, leave email where it is. Let email be email. It is store-and-forward. It's got no metadata protection. It's difficult to encrypt because it's always being added on top. And it confuses people. If people want secure messaging, let's go elsewhere. And the point is we've got clients. Clients are a dime a dozen. Ever since the Snowden revelations, alternative secure things, sort of generically, are being announced daily.

And so Matthew concludes by saying, okay, what's coming? There is great work being done that has nothing to do - oh, and I should mention that, even though Google and Yahoo! sort of have an OpenPGP feel, I haven't looked at it closely, but I've seen I think it was Matthew making comments that they're not fully PGP-compatible. So they're, like, adopting old protocols, but also not maintaining full compatibility. So they're sort of, again, creating something not what we want for today.

There is a protocol that Silent Circle is using called SCIMP, which is the Silent Circle Instant Messaging Protocol, where they've - basically they're saying we're not going to try to fix email. We're going to do a state-of-the-art protocol starting from scratch. And we've talked about Silent Circle. That's on its way.

DarkMail is the effort by Phil Zimmermann, who of course famously gave us PGP. He's not trying to keep PGP alive either. He's looking at how we move forward. He's working, famously, with Ladar Levison of Lavabit. And on their site, on the DarkMail.info site, they say: "Silent Circle and Lavabit are developing a new way to do email with end-to-end encryption. We welcome like-minded organizations to join our alliance." And so that's in the process.

Then there's something else which is moving along over on GitHub. Adam Caudill has something called SMIMP, which is Simple Messaging and Identity Management Protocol, which, if you look through the bullet points, I mean, it is everything we could want. And in fact the friend of the show Taylor Hornby, whose handle FireXware, has been involved in that effort and is given some credit for helping out with some of the security protocols.

And Adam writes that: "SMIMP is a communication and identity system designed to address the modern threats that weren't considered when the traditional email system was designed. Transparent encryption, forward secrecy, simple self-hosting, auditable user information, and strong privacy are all baked into the design from the beginning."

And not to leave something out, there's also another one called MailPile, which is MailPile.is. Unfortunately, it's built upon OpenPGP. So I really like the idea of, as you apparently do, too, Leo, starting over, coming up with a secure messaging platform. And I see nothing wrong with leaving email where it is. I mean, we don't have to replace it. We don't have to obsolete it. It has its place.

And one of the things, for example, I just scanned briefly the SMIMP page; but I noticed, for example, that there was a Proof of Work as part of, again, baked into the protocol. And of course that is antispam. You create a Proof of Work in order to make it expensive, relatively expensive, as opposed to, like, effectively zero cost to send somebody email. It's because email has zero cost that spam is a problem. If sending a message has a computational burden that is sort of proportional to your need to actually send it intentionally to another person, that creates a cost which prevents the whole spam mass emailing phenomenon.

So I do think it makes a lot of sense to look at a next-generation protocol rather than trying to burden our email system, which just, sure, if you have no alternative, there are ways to encrypt messages. But it just never got off the ground because it was always a

problem. And I think in this day and age, post-NSA, where our needs are significantly more mature than they were back then - for example, forward secrecy. No one's going to really feel comfortable knowing that their PGP or OpenPGP-encrypted email can be retroactively decrypted if they were ever to lose control of their keys.

So OpenPGP doesn't do it. PGP can't do that. It was never part of the protocol. I think it really does make a - and, again, it also doesn't hide any metadata. So somebody can always see that you're interacting with someone else. So we will be, in years coming, maybe even months coming, keeping our eyes on these developing protocols because it looks like some really smart people are focusing on solving this problem for us.

Leo: You know, we go full circle because the same problem really with email as with these photos in iCloud, the real key is to have encryption available to us for things that go out to the Internet.

Steve: Yes.

Leo: And means to keep them encrypted. Trust No One, you've always said this. Pre-Internet Encryption, PIE.

Steve: Yup.

Leo: And if you had encryption, you'd have control. The only way to control stuff in the cloud is to have Trust No One encryption.

Steve: Yeah. And unfortunately, Apple's model does create a promise of convenience. And it's more difficult to pre-encrypt everything. You lose some features. For example, we've talked about cloud storage providers that give you the convenience of web browser access to your cloud data. Well, for you to have web browser access to your cloud data, you need to either decrypt that in the browser, or they need to be able to decrypt it for you.

And the good news is, because now we're seeing it being feasible to do crypto in the browser, we're able to offer those features where before we really weren't. But, yeah, we basically, we just want to consider that somebody else's computer, which is the cloud, is just storing noise. Just absolutely pseudorandom nothing. And so if somebody did crack our accounts, all they're going to get is a bunch of noise that is meaningless to them.

Leo: Yeah. I think this is something we can solve.

Steve: I think we're on the way to it.

Leo: Especially because of public key crypto. You don't have a symmetric key. You really have a means to do this.

Steve: Right. And today's modern keys are so much smaller. I mean, they're so practical.

Leo: I love that. You should see the 2,048-bit key block that I was using with PGP.

Steve: Well, people think something's broken. It's like, oh, what...

Leo: What's that gobbledygook? Except now, with S/MIME, I've got an attachment, as I said, a .p7s attachment, and universally my - Henry just asked, "What is that attachment?" And "I can't open it." And "What am I supposed to do with it?" And no one knows about this stuff. This doesn't exist. We need something a little bit - I don't know.

Steve: Yeah, yeah. I think the right solution is don't ask email to do this.

Leo: No.

Steve: Create an alternative platform that was designed from scratch with state-of-the-art crypto for that purpose. And maybe there can be a gateway between them, or maybe not. Because, again, users have zero friction associated, from everything we're seeing, just the clients will be free, everything will be free. So if you want to hold a really super-secure private conversation, you do it over here. For stuff that doesn't matter, where you want infinite compatibility, and everybody has an email address, you do it with email.

Leo: As always, you've cut through to the gist of the matter, the nub of the matter. Thank you, Steve Gibson. GRC.com is the place he hangs his hat and his SpinRite. You've got a copy right there of the world's best file, hard drive, I should say, maintenance and recovery utility. You should also go there to get more information about Security Now!, including 16Kb audio for the bandwidth-impaired and full transcriptions for those who like to read along as they listen. Thank you, Elaine Farris and Steve, for making those possible. It's also a good place to go if you have a question or a comment because Steve doesn't do email. He has a feedback form. It's GRC.com/feedback. Next week, security permitting, we will do a Q&A episode.

Steve: And we should make a scheduling note that, due to the Apple announcement, the podcast has been scheduled at 1:30 rather than 1:00. So, and that's even...

Leo: So what we anticipate - yeah, that's even soft. We anticipate roughly two hours for Apple. 10 to noon we'll do the live coverage next Tuesday the 9th. Then we'll do MacBreak Weekly noon to 1:30, and get to you at 1:30, I hope. We'll certainly try to keep to that schedule, but one thing we don't control is Apple. So we'll do the best we can.

Steve: Yeah, yeah. And believe me, I will be...

Leo: He'll be watching.

Steve: ...panting and watching the whole morning, too.

Leo: Well, we have other shows after you, so we have to keep it all on a schedule.

Steve: Yup, yup.

Leo: Good. Thank you, Steve. And, yeah, I guess I won't be here the first Wednesday in October, or first Tuesday in October.

Steve: Oh, we've got all month. We've got all month.

Leo: We've got plenty of time. We'll talk about that...

Steve: Yeah, plenty of time.

Leo: Hey, thanks, Steve. We'll see you next time.

Steve: Thanks, Leo.

Leo: Bye-bye.