All the Usual Suspects

Description: This week we have all the usual suspects: governments regulating their citizenry, evolving Internet standards, some brilliant new attack mitigations and some new side-channel attacks, browsers responding to negligent certificate authorities, specious tracking lawsuits, flying device jailbreaking, more IoT tomfoolery, this week's horrifying Android vulnerability, more Vault 7 CIA WikiLeaks, a great tip about controlling the Internet through DNS - and even more! In other words, all of the usual suspects! (And two weeks until our annual Black Hat exploit extravaganza!)

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-619.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-619-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Yay! We're going to talk about all the things that happened in the last two weeks. Yeah, we missed last week's episode because of Independence Day. But we're back, and fortunately nothing went too horribly wrong. In fact, lo and behold, Steve's got high praise for Microsoft. They're going to do a really good thing. He talks about his favorite new thingy and a SpinRite recommendation from 40 years ago. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 619, recorded Tuesday, July 11th, 2017: All the Usual Suspects.

It's time for Security Now!, the show where we cover your security and privacy online with this fellow right here, the Explainer in Chief.

Steve Gibson: Yo, Leo.

Leo: Steven "Tiberius" Gibson. You're wearing glasses today.

Steve: I am. I've been wearing contact lenses for 45 years.

Leo: You've been deceiving us.

Steve: I'll never forget, I was riding in my best high school buddy's parents' car. He was driving. This is before - we had our licenses, but we didn't have our own cars in high school. And I turned to Scott, and I said, "So, should I get contacts?" And without skipping a beat he says, "Absolutely." And I said, "Oh, okay."

Leo: Wow, Scott had an opinion, apparently.

Steve: Yeah, he did. And I've been wearing hard contact lenses, the original...

Leo: Wait a minute. You never got soft lenses?

Steve: Nope, RGP, Rigid Gas Permeable contact lenses for 45 years.

Leo: Those you had to really get used to. My first ones were glass. And then I did get eventually the permeables, which were somewhat more comfortable. And the good news is that you don't throw them out. They last and last.

Steve: Yup.

Leo: You know, I now wear daily - you might find daily wear more comfortable. You don't ever clean them. You just throw them out.

Steve: Yeah. I'm just - I think I'm at the point now where, I mean, you know, I've given up on vanity. I have no hair left.

Leo: You look good in glasses. You look more intelligent.

Steve: Well, okay, I can use the...

Leo: You were like a leading man before. Now you're like a normal geek.

Steve: Oh. Wait a minute. That's what I prefer.

Leo: A smart guy.

Steve: Not sure that I want to lose the leading man look, if I can hold onto that for another couple years. Anyway, we are back after our missing week. And the good news is the world didn't spin out of control. And in one podcast I think we're going to be able to fit everything. In looking over what happened, I thought, okay. I'm just going to call this one "All the Usual Suspects" because that's what we have.

We've got governments regulating their citizenry, evolving Internet standards, some brilliant new attack mitigations and some new side channel attacks, browsers responding to negligent certificate authorities, specious tracking lawsuits, flying device jailbreaking, more IoT tomfoolery, this week's horrifying Android vulnerability, more Vault 7 CIA WikiLeaks, a great tip about controlling the Internet through DNS, and more. So in other words, all the usual suspects. That's pretty much the way this podcast has been rolling lately.

Leo: Wow, wow, wow.

Steve: So, yeah, lots of fun and interesting news. Oh, and we are starting our official two-week Black Hat exploit extravaganza countdown. Black Hat is - I think it's like the 21st through the 27th at the end of this month, so a couple weeks from now. And already we're beginning to see early reports. We'll cover some of them this week. No doubt we'll have some more next week. And I would imagine in the weeks following Black Hat, lots of more details. So in some cases, as is always the case, there's sort of some early leaks about what's going to be presented, but we lack some detail that we don't get until the actual presentation. So it's a good time of the year for Security Now!.

Leo: It is. Thank goodness we've got security - now.

Steve: So our Picture of the Week is just sort of a headshaker. It's, you know, we've seen these sites, which are credible, which solicit people to put their email address into the site, and the site will then check all of the disclosed leaked username/password databases.

Leo: Yeah, have I been pwned; right?

Steve: Exactly, exactly. And the problem with that is that, I mean, it is a sketchy thing to do. Now, arguably, people don't feel that their email address is super secret. But it's not a big reach to imagine a dialogue like this one which was - it's a spoofed dialogue. I hope so. I hope this doesn't actually exist. But it's a dialogue whose title says, "Is your credit card number in a hacker's database?" And then actually the English is pretty good except for the very end. So it then reads, "You can easily find out now! All you need to do is enter its information here, and we will scan thousands" - thousands, Leo - "of hacker databases..."

Leo: Thousands.

Steve: "...to see if any they have match yours." So they almost pulled it off, but there was a little English tripping over the feet there at the end. And then of course this has three fields, like THE three fields: credit card number, expiration date, and zip code. Actually, they didn't ask for the CVV, which I guess would have taken it to the final stage.

Leo: Yeah. What's your mother's maiden name again?

Steve: Yeah. Just want to verify.

Leo: Just want to make sure, yeah.

Steve: Oh, lord. So anyway, let's just hope nobody is actually being presented with that and filling it out because it's clearly...

Leo: The name of the window title kind of gives it away that it's a spoof. However, I'm not going to say it out loud.

Steve: It does, indeed. And I didn't.

Leo: Yeah. But I do feel like this probably does exist. And somebody in the chatroom is saying, well, who would be dumb enough to fall for that?

Steve: No, and that's it. That's exactly it. Unfortunately, we have to recognize the size of the sphere of users of the Internet today. And, I mean, Leo, you speak to them on the weekends often.

Leo: I still do. Well, but it's not just - and I don't want to say they're dumb.

Steve: No.

Leo: It's people in a hurry, people not paying attention, people who are not wary and aren't, as we are, alert to the possibilities of fraud.

Steve: Who knows what HTTP stands for?

Leo: Right, right.

Steve: We know what it stands for, but most people don't.

Leo: We shouldn't discriminate against people just because they're not, you know, geeks.

Steve: Not us, exactly, no. And I'm not. And that's the point, is that we are a minority. And there are going to be people who go, oh, I want to know if my credit card is in

hacker databases. What a nice, convenient service.

So, okay. Top of the news is that last Thursday - I mean, we've got a lot of top of the news, but we'll start with the W3C finally decided, after much debate, to add the EME, which we've discussed in the past, the Encrypted Media Extensions, to the formal HTML5 spec. Not everybody was happy.

Leo: Unh-unh.

Steve: Or, as Techdirt's Mike Masnick wrote the next day: "Tim Berners-Lee Sells Out His Creation."

Leo: Yeah. I was so surprised that it was Tim who did this, really.

Steve: Well, he's the director, and...

Leo: He was pretty outspoken that this, well, go ahead, yeah.

Steve: He was. He was. But, well, so there's two aspects of this that Mike focuses on. And I'm with him on one of the two. So he wrote, just to give you a sense for his position, he wrote: "For years now, we've discussed the various problems with the push, led by the MPAA [Motion Picture Association], but with some help from Netflix, to officially add DRM to the HTML5 standard. Now," he writes, "some will quibble even with that description, as supporters of this proposal insist that it's not actually adding DRM, but rather this 'Encrypted Media Extensions' is merely just a system by which DRM might be implemented." Mike writes...

Leo: Yeah, but why else would you do it?

Steve: Precisely. And as he says, "But that's a bunch of semantic hogwash."

Leo: Yeah, yeah.

Steve: "EME is bringing DRM directly into HTML," Mike writes, "and killing the dream of a truly open Internet. Instead," he says, "we get a functionally broken Internet." Which I'll argue with. He says, "Despite widespread protests and concerns about this, the W3C boss and inventor of the web Tim Berners-Lee has signed off on the proposal. Of course..."

Leo: So disappointing.

Steve: Yeah, "given the years of criticism over this, that signoff has come with a long and detailed defense of the decision." So there are many issues underlying this decision,

but there are two key ones that I want to discuss here - first of all, whether EME is necessary at all, and whether or not the W3C should have included a special protection for security researchers because that's what's missing. And as we've heard me say on this podcast so often, all of the evidence demonstrates the huge benefit to researchers being allowed to poke at security without fear of the DMCA being used to throttle that research.

So Mike doesn't want this at all, that is, doesn't want EME, Encrypted Media Extensions. My feeling is it was probably inevitable. The alternatives were no access, no web-based access to protected copyright content, or the requirement for a browser add-on plugin to implement proprietary protections. And we're seeing...

Leo: That's what we've been doing till now, by the way. That's how it's worked up to now.

Steve: Right. And we're trying to move away from the likes of Flash and Silverlight and so forth to more of a standards-based approach. Or the alternative, of course, would be a separate custom application published by each provider to deliver their own content. So to me, I mean, I don't use DRM. I've never copy protected anything that I've ever published. But my sense is that either, well, I think it's clear that the copyright holders are never going to give in, that is, they're not going to just say, okay, fine, and make their content available without protection. So the alternative is to make it more cumbersome to have that content online.

So essentially what this does is it creates a standard container and a standard means of invoking what's called the CDM, which is the Content Decryption Module. And so that remains a per-provider blob. But the EME, the Encrypted Media Extensions, makes that a uniform deliverable which ends up providing a seamless experience to the web user who wants to access DRM-protected content through their browser. So it expands the portal. It probably makes it stronger in the long term.

But the biggest problem from my standpoint is that they have punted on the DMCA provisions. That is, in fact, exactly what they wrote was: "We recommend organizations involved in DRM and EME implementations ensure proper security and privacy protection of their users. We also recommend that such organizations not use the anti-circumvention provisions of the Digital Millennium Copyright Act and similar laws around the world to prevent security and privacy research on the specification or on implementations. We invite them to adopt the proposed best practices for security guidelines, or some variation, intended to protect security and privacy researchers."

But, I mean, and this was one of the big issues surrounding this, that they just, as I said, they just punted on. They did not put that into the spec. So it is certainly possible that somebody who in the view of the MPAA or the RIAA, whomever was the content owner, decided they didn't like somebody doing research into the security of their CDM, their Content Decryption Module, could thwart that research.

Leo: And that's really where people feel that Tim sold out because he could have supported this very small exemption, and didn't even do that. And I think that's the sellout.

Steve: Yes, yes. And there I agree. Again, I think what we're seeing broadly is this

continual maturation of the Internet. And to me, we are using currently separate applications, for example, on iOS devices, you know, HBO Go and HBO Now and Showtime's got its own module, and everybody has their own. And this, moving forward, to me this feels like a means of integrating this and making the experience more seamless. And the only way we're going to ever get that content in a web browser is if there's either a plugin or a standard. And so, again, people don't have to view protected content through their browsers. No one's making them do it. But this at least provides a framework. So I can see that. But I really do wish that they had said, okay, we're going to do it, but we're going to protect researchers also. And, as you said, Leo, they chose not to.

A very nice piece of new research has gone public in the most recent OpenBSD snapshot. Version 6.1 of OpenBSD brings us KARL, K-A-R-L. Now, we have ASLR, which is always a tongue-twister, the often-spoken-of on this podcast Address Space Layout Randomization. And there's the kernel version of that, KASLR, Kernel Address Space Layout Randomization. Now, exclusively for the time being because the Linux team are drooling over this, we have KARL, Kernel Address Randomized Link, a little awkward acronym, but KARL is what we have. And it is very cool. Under KARL, that is, from v6.1 of OpenBSD on, every freshly booted kernel will initiate a background process to randomly relink and rebuild a next version of that same kernel which, once finished, will automatically be used at the next system boot. And I know from hearing you talk about it, Leo, when you were playing with, I think it was Linux...

Leo: Yeah.

Steve: ...and recompiling the kernel from scratch. And every time I do it I just shake my head. Actually, for me it's FreeBSD, it's how can this possibly work? Because the screen, I mean, it's like some crazy, I mean, they wouldn't even show it to you on a movie looking like that. No one would believe that it's actually doing something. It just scrolls endlessly, these ridiculously long command line streams of stuff going by. And every one of those is a source module being compiled into an object module, a .o file. And then, finally, they're all linked, all of those object modules are linked together to create an image, and that image is what's then stored on the non-volatile store, typically hard drive or SSD. And then that's loaded into memory at boot. That is the kernel.

Well, address space layout randomization, as we know, takes modules of the system. Like in the case of Windows they're DLLs. And so the entire kernel is a set of a few, not many, like maybe five or six depending, and device drivers also, but a few big blobs. And with relatively low granularity, for technical reasons, they randomly place those in physical memory so that the address of the blob is not known to an attacker because we know that one of, you know, as we've tightened down the screws on exploits, we've made it more difficult to, for example, for an attacker to run code in a data buffer. And we've done that by setting the data buffer to non-execute, the so-called NX bit, tells the chip you are not allowed to execute code. You can treat it as data, but not as instructions.

So to get around that, attackers said, okay, they would use this so-called Return-Oriented Programming, ROP, where they would find existing little snippets of code in the kernel and jump to it in order to - it's like, okay, if you won't let me run my own code, I'll knit together existing code, which has to be marked as executable because it is, and perform my nefarious deeds that way.

So the problem is that, if any function in the kernel leaks its own location, that is, for

example, say that there is a function which obtains some information which it holds in its own internal buffer, and it returns a pointer to the caller saying here's a pointer to the information you requested. Well, that pointer will be to the buffer, which reveals the location of the buffer to the caller and so, in essence, that leakage of the position of that function. And what that immediately does is then, for anyone who is knowledgeable about the way the OS is built, that reveals the relative position of all the functions and all the code in that one blob, in that whole module.

So what we've seen is exactly that kind of workaround for address space layout randomization, any clue that can be obtained. And in fact, even if you don't have a clue, the granularity is typically one in 256. So it's only a byte worth of granularity. So if you have a scattershot attack where malware is just going to guess where something is, it'll be wrong 255 out of 256 guesses, but it'll be right one out of 256 guesses, and that may be all it needs.

So what KARL does only now on OpenBSD is that final stage of kernel building, where all of those .o, the individual small object files, which are normally independently compiled, they still exist. But the final link phase is reperformed in the background with completely randomized order. So the kernel that results is no longer broken into multiple pieces and relocated with low granularity. Instead, it is a custom-built kernel with deliberately randomized offsets to every single object file that composes it. So it's far more anti-exploit granularity, and it's available now in OpenBSD.

And the Linux team, as I said before, is jealous. They're looking already at borrowing the idea. So it may be that in the not too distant future this could become a very powerful attack-resistant technology, available for Linux family operating systems, which is of course the most popular open source OS on the planet.

Leo: Yeah, and the good news is that linking step doesn't take that long. That's not the stuff you see scrolling by mostly. It's the compiling that takes forever.

Steve: Correct.

Leo: So just relinking it, that's the only way you could even contemplate doing this. It probably wouldn't be, especially just if you show it on the screen, it'd probably take a few seconds, I would think.

Steve: Well, and so what they've done is they spawn a background rebuilding thread.

Leo: There you go, yeah.

Steve: And so if you happen to reboot before it's done, it just reuses the same one. But as long as it's allowed to finish, then it stages that rebuilt one, which was rebuilt probably using completely system idle time so that it doesn't have any foreground impact at all. It just uses that next time you reboot. So essentially...

Leo: That's clever. I think that's a great idea.

Steve: Oh, it's brilliant, yeah. I mean, it's like, okay, that's the way we're all going to be doing this in a few years because - if you can. I don't know how Microsoft can do it. There was some talk of adopting this in Windows.

Leo: Oh, no. You get a blob with Windows. I don't think there's any object files on the drive.

Steve: Exactly. That's the problem is Microsoft does not want you messing around with the Windows kernel. But for the open platforms, I think this makes a lot of sense. I'm bullish about it. It's very clever.

So, as I said, Black Hat 2017 is approaching at the end of the month. It was one year ago during the Black Hat 2016 conference that Apple's head of security, Ivan Krstic, announced with a great deal of fanfare that Apple would finally be joining all of the other large publishers, which at the time included Microsoft, Google, and Facebook, as well as countless smaller firms. And in fact, Leo, down toward the end of this I've got - you'll see a www.bugcrowd.com link. It's a really nice site showing all of the software publishers of all stripes that offer bug bounties - attention, acclamation, cash awards...

Leo: They want money.

Steve: ...hall of fame and so forth. So last year Apple said, "Us, too." And in Ivan's Black Hat presentation they detailed, or he detailed, Apple's five broad categories of bugs and their exploits. $200,000 would be paid in return for the disclosure of any flaws in iOS's secure boot firmware components. $100,000 for a vulnerability allowing the extraction of confidential material protected by the secure enclave processor. $50,000 for the execution of arbitrary code with kernel privileges, or for unauthorized access to iCloud account data on Apple servers. And, finally, $25,000 for demonstrating access from a sandboxed process to user data residing outside of the sandbox.

So now here we are nearly one year later, approaching the subsequent Black Hat conference. And by any objective measure, Apple's bug bounty program has been an utter failure. Why?

Leo: We kind of expected that.

Steve: We did. In fact, we anticipated it a year ago when they announced this. Motherboard did some research. They promised a handful of well-known exploiters their anonymity, and they also spoke to some others who were bound by Apple NDA, the non-disclosure agreements. They promised them anonymity in order to speak off the record. Everyone told the same story. Apple bugs are so rare and so difficult to find and are thus so valuable that no one who is looking for a payday from their efforts would turn an uncovered, high-quality exploit over to Apple. Why? Apple doesn't pay enough.

Leo: Economics, yeah.

Steve: The going rate, get this, the going rate on the gray market for a multi-exploit iOS

jailbreak is currently $1.5 million. And even second-tier exploit purchasers will pay half a million U.S. dollars for similar exploits. So it's like, eh, $200,000? Thanks, Apple, but if I can get $1.5 million? Yikes.

Leo: Right. Apple created this, really, by being so secure and such a good target; right?

Steve: Yeah.

Leo: And they for a long time didn't have any bug bounties because they said we don't want to get in a bidding war.

Steve: Well, yeah, exactly. And remember when we'd been covering the Pwn2Own competitions in Canada, sadly, my favorite browser, Firefox, kind of fell off the list of competition because it was regarded as too easy. And so, no, sorry, we're not even going to pay anything for Firefox exploits. Chrome, yes. IE, well, Edge, yes. Firefox, eh, don't bother. So, I mean, it really is what we see is the case, that the harder it is to beat, the more you are rewarded if you beat it. And so ostensibly the people who pay, you can imagine, I mean, the point is, imagine this. A purchaser who pays a hacker $1.5 million, that's profitable for them.

Leo: There's a reason they're paying that much.

Steve: Exactly. They're able, I mean, they have purchasers who are going to make that a profitable acquisition for them. Yow. So that's the world we live in today. Incredible. And the good news is that BugCrowd.com site demonstrates that there is now an ecosystem. Again, no one is paying this much money. This is up, you know, top-tier, high-quality, reproducible. This is what state actors and intelligence services, government level, who have a need to get into somebody's phone, they'll pay the money. And of course, famously, Apple has refused to help in some cases. For example, in the case of the FBI they refused to crack an iPhone open. And so apparently the FBI was able to purchase that from someone for some serious coin.

Leo: Looking at all these huge number of bug bounties, I'm guessing that people make a living doing this.

Steve: Yeah. And if I didn't have anything better to do, I think it would be kind of fun to see if I could find exploits and then responsibly report them. It's like, okay, you know, fix this. And this is why we have to not criminalize that activity. You could criminalize it being irresponsibly handled. That's arguably hurting the ecosystem. But don't criminalize responsible disclosure of problems found. We have to keep that alive.

The wheels of browser behavior turn slowly, but they eventually do. Two years ago, in July of 2015, two years ago, here we are in July of 2017, a significant breach of certificate authority policy was discovered by a user of WoSign's poorly designed free certificate service, who obtained a certificate for the entire GitHub root domain after only proving control over a subdomain. We covered the story at the time. And as we know,

GitHub's architecture, by its nature, gives individuals control over their own GitHub subdomains. That's what it's all about.

So in another implicit failure to abide by the CA rules, WoSign either did not detect or did not report and also did not revoke the misissued certificate. So this demonstrated no effective auditing and disclosure was present or performed, and that's part of being a CA. So this breach went undisclosed until a year later a British Mozilla developer discovered - actually it was 13 months later, and so it's 11 months ago that this had happened. He posted this on Mozilla's security policy mailing list, saying, "In June of 2015" - referring to this original breach - "an applicant found a problem with WoSign's free certificate service, which allowed them to get a certificate for the base domain if they were able to prove control over a subdomain."

Well, the original discovery was made, and we discussed this at the time, made using ucf.edu certificates and was then retested by the person who discovered it using GitHub. The person who discovered it responsibly reported the issue to WoSign, who revoked the reported certificate. But that person never reported the ucf.edu certificates, only the GitHub, like the GitHub subsequent misissuance. So a year later the originally misissued ucf.edu certificate had still not been revoked, further demonstrating WoSign's, as I put at the time, and again, "woeful" lack of responsibility. So during subsequent investigations in collaboration with Mozilla, Google conducted a public investigation which uncovered additional cases of WoSign misissuance of certificates.

So last October Chrome 56 began gradually reducing its trust of certificates signed by WoSign and its StartCom subsidiary. And so what they did was they blacklisted WoSign and StartCom, but then made whitelisted exclusions to permit certain sets of certificates, I mean, they didn't want to just lower the boom. They announced this is going to be the policy. And so they used date-based trust to continue to trust, based on when certificates were issued, the ranges that they thought had probably not been exploited. And but what they've done is they've said, but, you know, we have clear proof now that WoSign and StartCom are not honoring the obligations that come with essentially the ability to print money, as we've discussed in the past. Yes, making certificates is charging for bit patterns, but with it comes responsibility.

So as of or with the upcoming release of Chrome 61, all of that conditional trust of WoSign and StartCom certs will be terminated. Based on the Chromium development calendar, this change should be visible in the Chrome Dev channel within a few weeks, the Chrome Beta channel around the end of this month, July 2017, and then will be released into the Stable channel somewhere around the middle of September 2017. No more WoSign, no more StartCom accepted by the majority web browser on the Internet today, which is Chrome.

And as we know, a similar march is underway for Symantec, who was found to similarly not be living up to their bit printing and charging responsibilities under the CAB guidelines. So it always moves slowly, and I think it should. No one wants to make a mistake here. But there has to be enforcement of the responsibility that goes along with this.

So the most often-requested feature for Let's Encrypt is coming at the beginning of next year. January 2018, Let's Encrypt will begin offering, for the first time ever, wildcard certificates.

Leo: What? Fantastic. Because I had to spend some money, a lot of it, for wildcard.

That's great.

Steve: Yeah. From its inception, Let's Encrypt has been carefully, and we've been following along, rolling forward, trying hard, desperately hard not to make any big mistakes with something as critical as fully automated certificate issuance. They've held off on the much-requested feature of issuing a single certificate for all of a domain's subdomains, for example, *.example.com, or like *.github.com, where that certificate would protect any machines within that domain or any domains within that domain until they felt sufficiently safe, that is, until Let's Encrypt felt sufficiently safe and secure in offering wildcards.

So in January, this coming January, rather than requiring, as they do now and have, every subdomain to individually approve and obtain its own certificate, the single parent domain will be empowered to obtain a single certificate for all possible subdomains. Since the only thing Let's Encrypt automation is verifying is domain control, these certificates will be and can only be the least rigorous form of DV (Domain Validation) certs. But that's all any site needs, if it just wants to protect, encrypt, and authenticate its traffic. So for the majority of sites that didn't need to bother, like may have wanted to have HTTPS connections, but didn't want to get on the paid-for bandwagon, and may have been using free services like WoSign and StartCom were offering, well, those won't work soon at all under Chrome.

Let's Encrypt is an increasingly useful option. I still want Extended Validation. There are people who want organization-level, OV (Organization Validation) certs. For that, you still need to use a certificate authority like my favorite, DigiCert. But there are applications where you just want to encrypt your traffic. I mean, you can't securely log a user on, as we know, without HTTPS because, if you 'log on,' unquote, over HTTP, that state cookie, the cookie which is then associated with your logged-on-ness, is exploitable. It's in the clear. It's back in the Firesheep days that's what was going on. So you can't have any kind of security, and the notion of being logged onto something unless you're able to create an encrypted tunnel between the user and the web server. And that requires TLS, and that needs a certificate. So bravo for this. That's a nice move forward.

And I have to grumble again about headlines that have too much hook in them and not enough truth. Because the headlines read: "Researchers Crack 1024-bit RSA Encryption in GnuPG Crypto Library." Except, no, they didn't. The term "cracking encryption" means something, and that would be huge news if it were true. But it's not. What "cracking encryption" does not mean is arranging to obtain the private key by exploiting an algorithmic implementation flaw in a particular library. Yes, that's not a good thing, but neither is it a crack of the encryption. What did happen is that - and props to these guys because they basically pointed out and demonstrated a long-known problem that crypto library authors had been ignoring. They can ignore it no longer.

So this international group of eight crypto researchers used a flawed implementation of RSA to powerfully demonstrate that the underlying Libgcrypt crypto library, which is used by GPG, needs to be fixed. This implementation flaw was used to leak a private key - which is not a phrase you ever want to hear, you don't want your private key leaked - through a side channel. It used the Level 3 cache. And we talked about caching not too long ago, how L1 and L2 are in the chip or, like, in the core. And Level 3 is the larger cache just before main memory. So they used cache attack timing in a side-channel attack, which requires an attacker to run carefully designed and instrumented forensic software on the same machine where the private RSA key is being used, while it's being used.

So again, this is not a crack of 1024-bit RSA. This is a known implementation problem was vividly demonstrated. In the paper's abstract they wrote - and Bernstein was the lead author in this, Dan Bernstein, whom we talk about often, I know. He's the creator of the curve, as I've said, that was the inspiration behind SQRL. They write: "It is well known" - and I'm not going to go into the details, but this gives you a better sense for it - "well known that constant-time implementations of modular exponentiation cannot use sliding windows." Well, Leo, everybody knows that.

Leo: I thought that was common knowledge.

Steve: And you have to be careful, as we know, which way you open the window because, if you open the window the wrong way, that's bad.

Leo: And sliding doors, that's...

Steve: Oh, my god. Yes, exactly. They wrote: "However, software libraries such as Libgcrypt" - shame on them - "used by GnuPG, continue nevertheless to use sliding windows. It is widely believed, even if the complete pattern of squarings and multiplications is observed through a side-channel attack, the number of exponent bits leaked is not sufficient to carry out a full key-recovery attack against RSA. Specifically, 4-bit sliding windows leak only 40% of the bits, and 5-bit sliding windows leak only 33% of the bits." Which, you know, that would seem like a problem.

"In this paper," they write, "we demonstrate a complete break of RSA-1024 as implemented in Libgcrypt. Our attack makes essential use of the fact that Libgcrypt uses the left-to-right method for computing the sliding window expansion. We show for the first time that the direction of the encoding matters. The pattern of squarings and multiplications in left-to-right sliding windows leaks significantly more information about the exponent than right-to-left sliding windows. We show how to extend the Heninger-Shacham algorithm for partial key recovery to make use of this information and obtain a very efficient full key recovery for RSA-1024." And then they note: "For RSA-2048, our attack is efficient for 13% of possible keys."

So there you go, Leo. You've got to be very careful of which way you slide the window, and you don't want to leave it open by mistake because you could lose some bits. Anyway, so don't worry, RSA-1024 is intact. And I am sure that this will represent a lesson learned by the Libgcrypt folks, who will probably fix their code, and we'll get an update...

Leo: Close their sliding windows.

Steve: They're going to slide the window in the proper direction because they weren't. And leave it to Dan to point that out.

Leo: By the way, so I've been using 4096-bit keys. When they say 1024, are they talking about key size?

Steve: Yes, the length of the private key [crosstalk].

Leo: So if you used a 2048 or 4096, this wouldn't impact you no matter what lib you used.

Steve: Well, 13% of 2048-bit keys could be efficiently recovered.

Leo: Okay.

Steve: And they didn't talk about 4096. My guess is that the percentage of leakage would probably be fixed. But you've got so many excess bits in there, Leo, that you could lose half of them, and nobody would - it wouldn't help anybody enough.

Leo: I'm realizing that might have been a mistake because I'd have to use - I can only use the YubiKey 4 with that size of a PGP key. They don't support the 4096.

Steve: Yes, luckily we're not at the 8Tb private key point at this point. So I thought this was an interesting little piece. The good news is a judge has his head on right. There was a lawsuit which a bunch of people have been continually trying to bring against Facebook for five years. The Guardian reported last Monday that a judge had dismissed a lawsuit, this time with prejudice, meaning that a subset of the claims cannot be brought up again, which was accusing Facebook of tracking users' web browsing activity even after they logged out of Facebook. The plaintiffs in this suit alleged that Facebook was using the Like buttons found on other websites to track which sites they visited - duh - allowing Facebook to assemble detailed records of their browsing. Yes. That's what they're for. That's what they do. The plaintiffs argued that this violated federal and state privacy and wiretapping laws.

However, U.S. District Judge Edward Davila in San Jose, California, dismissed the case because he said that the plaintiffs failed to show they had a reasonable expectation of privacy or that they suffered any realistic economic harm or loss. In other words, the plaintiffs were annoyed, but annoyance is not sufficient grounds for a lawsuit. Davila said the plaintiffs could have taken steps to keep their browsing histories private, for example, using the Digital Advertising Alliance's opt-out tool or using incognito mode, which is now available in all major browsers. And they failed to show that Facebook illegally "intercepted," which is apparently what they were alleging, or eavesdropped on any communications.

He wrote: "Facebook's intrusion could have easily been blocked, but plaintiffs chose not to do so." Davila also dismissed an earlier version of the five-year-old case back in October of 2015. He wrote: "The fact that a user's web browser automatically sends the same information to both parties" - meaning the site you're visiting and the site whose assets are also hosted on that page - "does not establish that one party intercepted the user's communication with the other." He said: "The plaintiffs cannot bring privacy and wiretapping claims again, but they could pursue a breach of contract claim if they chose."

So standing back from this, it sounds as thought the plaintiffs are unhappy that logging out of Facebook does not stop Facebook's tracking of their activities. But I think they need a lesson in how the Internet works. As we know, websites give a browser a unique

token, a cookie, which is subsequently returned anytime that browser makes any request for any asset from that - in this case Facebook - domain. But that's entirely separate from logging in. Even if you never were to log into Facebook, just by touching the Facebook domain, which your browser would do anytime you visited any page on the Internet which contained a Like button, Facebook would still give your browser a cookie, which could then be used to begin assembling an anonymous profile of essentially you, anonymously, as your browser, because it gets uniquely tagged, even if they have no idea who you are.

Just going anywhere and picking up that cookie then allows that to begin. You don't even have ever needed to visit Facebook ever, since any contact with any page makes that happen. And then of course later, if you did visit Facebook and logged in and created an account and associated yourself with that Facebook session, then all of the pieces suddenly click together and fit together. But all logging in is doing is telling Facebook that that cookie which has been wandering around and appearing every time your browser came into contact with any page having a Facebook Like button, all logging in is doing is setting a flag saying this user is logged in. And when you log out, that flag is reset. And the resetting of that flag just means don't let this person do anything on Facebook until, like, only give them the login page until they do log in, and then they have freedom to do whatever they want to on their page. So anyway, I'm happy to see that this didn't go any further, and these people just need to know how the Internet works.

In our continuing "when governments react" watch, we have China upping the ante on VPNs. They have instructed the three major carriers of content in China, which is China Mobile, China Unicom, and China Telecom, to formally block all use of VPNs by February of next year, February 2018. And so this is sort of a - VPNs have been pushed against by the Chinese government to the point that already the Android Store has lost pretty much all of the Chinese-based VPN services. They're just no longer available there.

And as we know, we've talked about the so-called Great Firewall a number of times in the past. China has been blocking content that they don't want their citizenry to have access to, including Facebook, Twitter, YouTube, and Instagram. News sources like New York Times and Wall Street Journal are blocked, as well as Google Scholar. And as we covered on the podcast at the time seven years ago, Google's own reluctance to have its results censored led it to quit the country, and then the Chinese government subsequently banned most of Google's services.

So a way around all of this border filtering with the Great Firewall has been the use of an encrypted tunnel that would mask the accesses that somebody using the VPN was pursuing. And so that had been an increasingly popular way, as people inside China wanted to get access to the whole Internet, to an unfiltered view of the Internet. China has said, okay, we're going to make that increasingly difficult. Aside from the policy angle, our question is, is it even possible? Once upon a time, as we know, old-school VPNs used well-known ports. They used specific ports, and that's how some corporations and public WiFi setups and so forth would deliberately block the use of VPNs when they wanted to prohibit it.

Modern VPNs, though, are able to tunnel over TCP or UDP and have complete flexibility of which ports they use. Which makes filtering them much more difficult. And, for example, it would be possible for a website to offer access to the rest of the web if it wanted to. So, for example, somebody in China could access a non-blocked website into which they put the URL they want to visit, and that site presents them with the page content from the URL they provided. You could sort of think of it, anyone who's ever used like the web archive has seen something like that, where you go to the web archive site, and you give it the URL you want to look up in the past, and it shows you the page that was archived.

Well, it could as easily be done with live content from the web on the fly. Essentially, you're sort of using that intermediate website as a deliberate man in the middle to perform the accesses for you to content that would otherwise be filtered. And, yes, now then you end up with a cat-and-mouse race where it's necessary to go stomping out all of the sites that are offering a service like that. I just, you know, it just seems infeasible. So I think what this means is that the convenience and ubiquity of using easier-to-use VPNs will probably get throttled. But it just doesn't seem that there's a practical means for really blocking somebody who's determined to access external content to do so.

On the other hand, the convenience of having a Twitter app that just watches the stream, that you'll lose because you would really need a traditional VPN in order to do that. But again, we're watching governments and state actors deciding that they want to profile their population's use of the Internet to suit their own needs, increasingly, as the Internet becomes more of a thing. And believe it or not, drone jailbreaking is now a thing.

Leo: Finally.

Steve: DJI, the maker, as we know, of what is arguably the best line of commercial quad copters, especially useful for professional aerial photography used in weddings and bike races and especially watching Apple assemble their solar doughnut, is locking down their drones against a growing army of DIY hackers who are arguing for freedom of flight. This controversial DJI firmware has for some time been enforcing no-fly zones and also altitude and air speed controls, which upset drone owners who chafe at such limitations. And online boards are full of reports where the restrictions appear to be applied arbitrarily, and in some cases improperly, apparently restricting flight for no good valid reason.

So in response, hacking DJI's drone firmware has been on the rise. And DJI has started to fight back by pushing firmware updates to break the jailbreaks to Internet-connected drones, and they've been proactively removing previously available legacy versions of their own firmware from their servers to prevent people from getting access to it and retrofitting older firmware which can then be jailbroken and used. So what of course has been spawned is an underground of firmware archives and so-called "chop shops" which are retrofitting full flight freedom firmware.

And the legal ground, sort of the legal position is a little bit unclear in the U.S. because, once again, the Digital Millennium Copyright Act is present. The Librarian of Congress, which administers specific exemptions to the DMCA, has given so far wide latitude to tinkerers who seek to break through software locks for the sake of repair or restoring factory defaults. So, for example, it's currently legal to hack into trailers, cars - I'm sorry, tractors, cars, and cell phones, but not legal to jailbreak videogame consoles. But there's currently no specific exemption for drones, although I guess DJI would have to bring a lawsuit against its consumers in order to test this and establish some precedent.

But so now we have a - I doubt that, without really upping the stakes, DJI is going to be successful. We've seen, for example, what Android mobile devices and iOS mobile devices have had to go through in order to sufficiently strengthen themselves to the point that it's becoming, you know, it's worth $1.5 million to get a jailbroken iPhone exploit. So anyway, I just thought it was interesting that there's now another jailbreaking front in the case of people wanting drone freedom.

And, okay, in this week's horrific Android vulnerability, we have Broadpwn, which we

don't know everything about, and we won't for two weeks, until Black Hat. But we know a lot. First of all, it's very worrisome, affecting, I mean, I don't even know the number, at least millions of Android and probably iOS devices. Unpatched Broadcom WiFi chips, okay, so not cellular, the WiFi chip, which are used in both Android and iOS devices, are vulnerable to a bug that allows an attacker to execute code on those devices without any interaction needed from the user and, even worse, simply by being within radio range of any malicious WiFi access point. You don't even have to join it. You don't have to do anything. Your mobile device just has to receive its signal.

A security researcher named Nitay Artenstein or "steen," I never know, no one knows, discovered and nicknamed the Broadcom firmware flaw Broadpwn, P-W-N. It's got a CVE number, 2017-9417. And he will be delivering the full presentation about Broadpwn at this year's Black Hat USA security conference at the end of the month. He responsibly disclosed the bug to Google, who already included it in the Android fix which was available in last week's July 5th Android security update. So for devices that are receiving Android security updates, you got fixed last week.

Although little public information is available yet, we know, and Artenstein has said, that Broadpwn affects millions of Android and iOS devices that use Broadcom's WiFi chips. The flaw is present in the firmware for a huge and old line of Broadcom. It's the BCM4300 family, or 43xx family of WiFi chips, which are included in an extraordinary range of mobile devices, including vendors Google with the Nexus line, Samsung, HTC, and LG.

And then, after being made curious about this, another Android security expert reverse-engineered last week's July security patch from Google to dig out more details about Broadpwn. He determined that the bug appears to be a heap overflow in the Broadcom firmware. The researcher said that the exploitation occurs when the user's device receives a WME, which is a quality-of-service information element, with a malformed length from any network. And as I noted, exploitation does not require user interaction. The victim needs only to enter into the WiFi range of an attacker's signal. Artenstein has confirmed that even connecting to a malicious network is not necessary. Not surprisingly, Google's security bulletin last week rated Broadpwn as critical.

This BCM43xx line appears to date back at least 12 years, to 2005. It's unclear whether this flaw was present back then. But as we've discussed on this podcast previously, this baseband OS is a concern because it doesn't get much attention. We're always talking about iOS or Android, the visible OSes that we see and that our apps interact with. But there is this underlay OS that runs the cellular and the WiFi that performs the communication. And it's firmware-based, too. And it tends not to change nearly as often as the OSes that run on top of it because it's kind of part of the lower level hardware offering. You know, it comes from Broadcom with the chip.

So it's very clear that, if this flaw were to be weaponized by any global state actor and subject to any exploitation limitations, that is to say, we don't know for sure what exactly you can do with this. So you can run code, what, on the Broadcom chip? Well, what does that mean? We don't really know yet. But that means that virtually any unpatched Android device could be compromised simply by getting a malicious portable WiFi access point within WiFi range of any targeted Android user.

We keep covering these serious Android problems. In older and unpatched devices, they are rapidly piling up, that is, these problems are. And I would argue they've already far exceeded critical mass. Although targeted exploitation is unlikely to affect many of us, opportunistic exploitation on college campuses and in public settings such as airports, hotels, or <gulp> the upcoming Black Hat conference could easily affect huge numbers of

users, virtually anybody with an Android device with their WiFi radio turned on.

So the conclusion to be drawn from this as we step back a bit is clear. Android devices that are not continually being rapidly and responsibly patched cannot be used securely. And unfortunately, depending upon how far back this goes, there is a massive population of devices that are not being patched, that have been abandoned, but are still in use. And if this is exploitable, that is, if the exploitation of the baseband OS firmware can be turned into something, it's a goldmine for the weaponization of this against old devices that are no longer being patched. We have, as I said, no information at this point on the status of this for iOS devices. We'll see what Artenstein says in two weeks during his Black Hat talk.

Leo: And learn how he pronounces his name.

Steve: Yes. I hope he says.

Leo: Very exciting.

Steve: So our listeners know that I have a somewhat contentious relationship with Microsoft. I'm annoyed when they do things, you know. I mean, I wrote Never10, which I looked yesterday: 2,653,204 downloads. And it's about 2,000 more a day.

Leo: That's awesome. What, still?

Steve: Yeah.

Leo: Okay.

Steve: Yeah. But I do want to give them props when they do something that I think is really great. Doesn't happen that often. Yes, I know. But they've done something that I think is fabulous: a new form of very clever selective whitelisting will be coming to Windows 10. It's called Controlled Folder Access. It will be debuting in September's Fall Creators Update. And Win10 testers now have access to a preview of the changes, which will include this new Controlled Folder Access, or CFA, feature. CFA is designed - and this is what I think is just Microsoft nailed this one - to only allow specific apps to access and read and write to specific folders. If enabled, the default list prevents unknown and unauthorized apps - read "cryptomalware" - from accessing the desktop, pictures, movies, and documents folders.

Leo: It's funny because that's how iOS works; right?

Steve: Correct.

Leo: Yeah.

Steve: But, you know, Windows has been just a free-for-all. So in the new version that is coming, CFA can be enabled. The dialogue is the Windows Defender Security Center. There's a Protected Folders option and an "Allow an App through Controlled Folder Access." So I think this is a brilliant tradeoff. We've talked before often about how blacklisting is prone to failure. I mean, because you're saying, oh, don't allow this, don't allow that. So something just pretends to be something else. So the only way to protect yourself is a Deny All and then Allow Selected policy.

What's not clear is how this will work interactively, that is, whether or not, first of all, it'll be enabled by default, whether it might start off being disabled and then be enabled in the future once Microsoft gains more experience with it, what programs will be permitted, et cetera. But as we've been saying here again, whitelisting, a default block and selective permit is the only way to be truly secure. And of course you have to also check that an application is actually what it says it is so that you just didn't have malware pretending to be WinWord and then allowing it access to your documents folder.

But bravo. There is an impact on the user because they may be a little confused. But I think you could work around that. If they downloaded some new program that wanted access to their documents, then you'd have to permit, manually permit that program to have access. But what that would mean is that malware could not surreptitiously slip into your system and then surprise you with a big scary red screen telling you what percentage of a bitcoin you have to pay in order to get your stuff back.

So bravo. This is the kind of movement we need in order to tighten things down. And I think what's clever about it is that it's constrained. That is, it would break everything if Microsoft tried to establish wholesale whitelisting policy across the entire operating system. Instead, they're just saying these are the folders where most of users' important stuff is. And the universe of applications that need access to that subset of folders is reasonably small. So we can constrain the access to that subset of the whole OS. And the universe of apps that need access to it, we can constrain that, too. So just, yay, props to Microsoft on that one.

In our This Week in Bizarre IoT Stories, we've had the smart water meter that contradicted somebody's story about the use of the hot tub in the middle of the night. Now we have Engadget reporting a somewhat bizarre piece of IoT news, that a smart home voice response device - and by the way, it was originally misreported as being a Google Home device, but we still don't know what make and model it was - was responsible for ending a violent dispute by calling the police. "Police in New Mexico reported that a smart home device intervened in a domestic violence incident by calling 911."

Leo: Oh, come on.

Steve: I'm not kidding. "When Eduardo Barros asked, apparently in a loud voice, 'Did you call the sheriffs?' as he threatened his girlfriend with a gun during..."

Leo: Oh, "call the sheriffs." I get it.

Steve: "...during a fight, the device interpreted it as a request to call emergency services. The 911 call responders overheard the altercation and called both negotiators and a SWAT team, who arrested Barros over assault, battery, and firearms charges after a stand-off. Barros's girlfriend was hurt in the altercation, although police contend that the situation would almost certainly have been much worse. County Sheriff Manuel Gonzales believes that the command 'possibly helped save a life,' including that of the girlfriend's daughter, who was thankfully unharmed."

Leo: We don't know what kind of device, though.

Steve: We don't. We know it's not a Google Home device, so that...

Leo: And Amazon has apparently said it's not them, either.

Steve: Interesting.

Leo: I don't know what it is.

Steve: So I know from firsthand experience, and I'm sure you do, Leo, because you've got one in every room now, and even in your closet...

Leo: Everywhere.

Steve: ...that unintended sounds, especially loud sounds, can cause the command discriminators in these devices to misfire. For some time I had an Amazon Echo in my living room, listening with me to my whole room theater. And from time to time, when nothing to my ear sounded like its wakeup trigger word, the device's blue ring would suddenly illuminate, scan around a bit, and then it would go back to sleep. And I used to kind of look at it going, okay. So it's entirely believable that this could happen upon hearing especially a shouted, you know, screaming voice, "Did you call the sheriffs?" Especially if the girlfriend's name was you know who.

Leo: Oh.

Steve: So, yeah.

Leo: If your name is A-L-E-X-A, you're safe. That's funny.

Steve: Also in two weeks at this increasingly anticipated Black Hat conference, we will have a full presentation of Skype & Type. A group of researchers have developed and proven yet another side-channel attack. Forbes reported in an advanced demonstration of the Black Hat presentation that, after sufficient training, which can happen in the background during and over a Skype call, a randomly chosen password was determined

within seconds using only information gleaned over the connection from the unique and distinctive sounds and timing of someone typing on a keyboard.

So we've talked before about the keyboard as a mechanical device, vibrations from typing being picked up by a cell phone lying near the keyboard. In this case the actual sound, especially my keyboard because it's an old clanky one, I imagine, the sound of the keyboard, even through Skype's compression and decompression of the audio, was sufficient to allow a random character password to be determined within seconds. Full details in two weeks at Black Hat.

WikiLeaks and the Vault 7 document leakages have last week informed us of another CIA, alleged CIA project known as OutlawCountry. It is a very powerful, but apparently also very targeted, remote monitoring kernel exploit affecting some Linux-based machines. It reportedly allows CIA hackers to redirect all outbound network traffic on a targeted computer to CIA-controlled computer systems for exfiltration and infiltration of data. It's a Linux kernel module loaded via shell access to the targeted system. So they need to have some way, some other means of getting it in. But once there it creates a hidden Netfilter table under an obscure name on a targeted machine.

However, it's not without limitations. The leaked documents indicate that OutlawCountry v1.0 contains one kernel module for the 64-bit CentOS/Red Hat v6.x version of Linux, stating that "This module will only work with default kernels," suggesting that the tool may have been developed for specific targeted use, where the intended target was known beforehand. But, yes, Linux systems, too, not just crazy SMBv1 exploits.

Somebody tweeted to me an interesting-looking Kickstarter project. And it started off looking like yet another maybe overhyped custom router. And we've talked about those before. You know, the CUJO comes to mind because they're alleging the ability to do, that is, in CUJO, things they can't do in a post-HTTP world, in a TLS world where all traffic is encrypted. First of all, this thing that was called Firewalla, F-I-R-E-W-A-L-L-A, I looked at that. I didn't dig in too deep, but I looked carefully at what they were claiming. And they did not appear to be overselling what it could do. That's the good news.

But what I was reminded of was something I've been meaning to talk about before. And two weeks ago, when I went off on my rant about Cisco overselling their "seeing into encrypted communications," which really looked like it reduced to using DNS metadata, I forgot to mention, and this reminded me, of something that some very good friends of mine and of the show have done. And these are the Nerds On Site folks, who were one of our very early podcast sponsors.

Leo: Dave's in the chatroom right now, I think.

Steve: Well, David Redekop is the person whose name is on the blog. What these guys created is something called DNSthingy.

Leo: Good name.

Steve: It's a great name, DNSthingy. DNSthingy.com is the site. And they nailed it. It is an affordable subscription service, less expensive than the other things that we've seen. And what they've done is they've fully thought through the power of DNS. That is, even in a land of encrypted communications, devices in your Intranet have to get the IP that

they're connecting to. As we know, by default, DNS is not encrypted. So this allows them to exert very granular per-device controls.

So, for example, the kids' iPads or computers could have time restrictions put on them so that they just stop being useful at bedtime. They can have family-safe content filtering put on them or not. The whole Internet can appear to be shut down or not. They have installable firmware for ASUS routers and a whole list of them that they support, or routers based on ClearOS or, my favorite, pfSense. So as we know, you can take a very inexpensive little standard router PC, load it up with pfSense, add DNSthingy to it. They've got a beautiful little dashboard for configuring it.

Anyway, we know who these guys are. They're 100% aboveboard. They're the real deal. Great people. And so I would commend our listeners to take a look, just check out DNSthingy.com. One quick way to get a sense for it is to go to the blog and scroll back down through David's postings because it's active. It's being maintained. They're all fired up about it. And I just think, I mean, it's the way to deal with a post-encryption world where you still want to apply per-device controls.

One thing you could do, for example, is constrain what your IoT devices can do. You could say to your camera, you are only allowed, camera, to look up the IP of the service that supports you, and nothing else. So if malware got in there, it couldn't go checking out command-and-control servers anywhere else. Its DNS queries would get blocked. Oh, and it does ad blocking, too. So you can say I want specific devices to be ad blocked or the whole household to have ad blocking.

Leo: Is it hardware?

Steve: No, it is software...

Leo: It's an update to your router.

Steve: Right, exactly.

Leo: Some routers do this, of course. In fact, the Eero just added this feature to their routers. So that's cool. So if your router doesn't do it - although even routers with access control lists, not a lot of them have ad blocking. That seems like a pretty - I don't think the Eero does ad blocking.

Steve: Well, and again, like they sat down and, as we know, they're tech-y. And they said, okay, what's everything we can do? So there's, like, time-based restrictions, per-device restrictions, and way more. So it's very capable. For a family router where Mom and Dad want to manage what the kids can do, you want content blocking. You want time restrictions. And, for example, you could also prevent somebody - I saw this at some point on the page or on the site. You can prevent someone from changing their DNS to something else in order, for example - the example they used was forcing OpenDNS to be used and prevent someone from changing it to something else in order to get around OpenDNS's restriction. So I've been meaning to talk about it, DNSthingy.com. These guys did a great job.

Leo: That looks cool. They do relocation, too. So you can access geographically restricted content. That's kind of neat.

Steve: Yeah.

Leo: Yeah, it's pretty sophisticated. Wow. Nice job, Dave.

Steve: Next week we're going to talk about something that Bruce Schneier referred to as, he said in his blog posting, "This is nice work." And I'm just going to share a couple juicy bits because we'll cover it in detail next week. But from the abstract they wrote: "We present the password reset MITM attack and show how it can be used to take over user accounts. The attacker initiates a password reset process with a website and forwards every challenge to the victim, who either wishes to register in the attacking site or to access a particular resource on it. The attack has several variants, including exploitation of a password reset process that relies on the victim's mobile phone either using SMS or phone call."

They write: "We evaluated [what they called] the PRMITM attack on Google and Facebook users in several experiments and found that their password reset process is vulnerable to this password reset man-in-the-middle attack. Since millions of accounts are currently vulnerable to the PRMITM attack, we also," they write, "present a list of recommendations for implementing and auditing the password reset process." We'll get into it in detail next week. There's a lot there. And as I have said recently, man-in-the-middle attacks and website spoofing, that remains the real challenge going forward to the Internet security because it's just so easy to get tricked.

There is a fabulous YouTube video for our little Miscellany here. I gave it a bit.ly shortcut because it was so good: bit.ly/sha-256. So again, bit.ly/sha-256. This guy does a brilliant job of helping us to visualize just what 2^256 means. He only made one mistake, early in the video, because he said that 2^256 meant that you had to guess an average of 2^256 guesses. Well, we know that's wrong. If you've made 2^256 unique guesses, that would give you a 100% chance of guessing it because that's all there are. So what he meant was 2^255 guesses would give you the average number required.

Leo: Half the set.

Steve: That is half that set, exactly. Other than that tiny mistake, he did a beautiful job. And so if anyone wants to get a - it's fun. It's only a little over five minutes long. But really it characterizes and puts into beautiful concrete context what 2^256 actually means. And that's important because that's how many bitcoin addresses there are, that's how many SQRL identities there are, and so on. So really nice piece of work: bit.ly/sha-256.

Also I wanted to note, and I've got the links here in the show notes, there are two Security Now! T-shirts that someone created. One is a very nice TNO Trust No One T-shirt, and the other is a T-shirt based on the "S" in IoT joke, you know, the "S" in IoT stands for security. Anyway, they are at Teespring.com, T-E-E-S-P-R-I-N-G dotcom.

I did have a little quick note about SpinRite. Bill Taroli apparently was having a problem.

Looks like he's a Linux user using the Btrfs file system. And so his first tweet said: "Sigh. One offline uncorrectable read error on my SSD." So as we know, SSDs fail, too. So he tweeted: "Time for a Level 2 scan with," and then he said "@GibsonResearch @SpinRite," and then did a "#securitynow."

Then a couple tweets later, it was some time later, he tweeted again, replying to the previous tweet. And he said: "@SpinRite awesome! SMART says drive is 63% useful life, but it feels so much faster now. Full Btrfs defrag under 30% iowait now." So Bill is obviously scrutinizing the operation of his system closely. He's looking at the iowait time that his Linux Btrfs file system is experiencing when he's defragging it. And after running SpinRite on his SSD, he saw the iowait time drop dramatically because just running a Level 2 scan with SpinRite over the SSD cleaned up the SSD and sped it up significantly.

And then I saw one more thing that really brought kind of almost a tear to my eye, Leo, and that was Chris Erickson tweeted, and I have the link in the show notes, but then there's also two pictures on the next page: BYTE magazine, November 1988. And the first picture is the cover of BYTE magazine showing Steve Jobs's new machine for the '90s, the NeXT, the NeXT computer. And then, under reviews, next to the big word "BYTE," it says - it's reviewing a number of things, and in there is SpinRite. And it was sort of, for me, a blast from the past. But as you know, Leo, because we were active at the time, BYTE magazine, it was the magazine of record back in the day, you know, in the late '80s. I mean, it was the bible.

And so Richard Grehan was the BYTE senior technical editor at large. And he finished his review - this is just the last two paragraphs. He said, he titled it: "A Must-Use Utility." "All I can say," he wrote, "about the Gibson Research people is that they did their homework. The user interface is well thought out and easy to use; all interaction is via an easy-to-navigate windows system. The package comes with" - now, he said a hard disk, but he meant a floppy disk - "and a 40-page user's manual that is more interesting for its historical content, how the authors of the package" - and we know that the author is singular - "made all their discoveries about hard disks, than any other information. The program is so well put together," he wrote, "I found I seldom referred to the manual anyway."

He said: "SpinRite is no 14-disk grand slam C compiler, but you shouldn't underestimate its usefulness. If you have a PC with a hard disk drive that you spend most of your day relating to, and your heart sinks every time you see the drive's bad sector list, SpinRite is what the word 'must' was invented for." And, oh, my god. Can you imagine? I mean, I just - I memorized that review.

Leo: Does he say anything about how he's waiting for v6, or no?

Steve: No. That was v1.0, I think, back then.

Leo: Yeah, sure, yeah.

Steve: Yeah, wow.

Leo: You must be one of the long, if not the longest running program still in

existence.

Steve: Thirty-nine years, my friend.

Leo: That's got to be a record.

Steve: It will be 40 years in 2018.

Leo: Well, we're throwing a party for that one.

Steve: Yeah, wow.

Leo: Wow.

Steve: Two quickie "closing the loops" from our listeners. Ian tweeted from @ianc. He said: "@SGgrc Listener for the past year. You often talk of prime numbers. Other than basic knowledge of prime, why are they so powerful for crypto?" And I'll answer quickly, just say that many types of cryptography are based on a problem known sort of generically as a trapdoor function, the idea being that it's something that's easy to do, but difficult to undo. And the only reason that prime numbers ever come up is that the most venerable public key crypto, that is, asymmetric crypto, where you are able to openly publish one of the keys to use for verifying a signature or verifying a cryptography, but keep the other one secret, is that it is trivial to multiply two prime numbers. And we have never yet found a fast way of demultiplying them, that is, of factoring the product of two primes.

That's the only thing, the only principle, the only reason primes are a deal is that it's easy to multiply, but the prime factorization problem has proved to be intractable for decades. And it's the reason we were talking two weeks ago about the whole "what does quantum mean" and the horror of an 8Tb public key. Essentially, a private key is one half of the public key. And so you take the private key as one of the primes and another prime. You multiply them. And so essentially, because you can't pull it apart, you're hiding the private key in plain sight. You're saying, here you go. Here's your public key. And what's cool about the RSA algorithm is it's able to use the unknown primeness of half of that public key to get some work done, but no one is able - it's only by knowing the half that is the private prime that you're able to take advantage of that. And that part you keep secret.

So that's, I mean, it's elegant in its simplicity, and it's so cool that it has remained a secret for so long. At some point we will talk in detail about elliptic curve technology, which is the newer system. It's the one that SQRL uses and that is becoming increasingly popular. The advantage of it is that, in order for prime factorization to be slow enough, the primes have to be so big that the resulting product of the two primes, the public key, is huge. And so the elliptic curve problem is a much harder problem to solve. Therefore, you get equivalent strength with a much smaller public key. And that's one of the - it just makes it much more convenient.

And lastly, someone who calls himself Acid Trucks on Twitter said: "@SGgrc What are your thoughts on using router features like parental controls to shield the Internet from IoT devices?" And I, coincidentally, already answered that when I was talking about DNSthingy and some of the clever ways that you could use blocking DNS or parental controls to keep somebody, especially an IoT device, from having unfettered access to the Internet. You just say no. Any query coming from a given MAC address only has permission to access the IP at a given domain. Otherwise, deny it.

Leo: That's interesting.

Steve: Yeah.

Leo: So DNSthingy would do that.

Steve: Yeah.

Leo: That's a simple way to do it. Just say, hey, this device, unh-unh-unh.

Steve: Exactly.

Leo: Nice.

Steve: You can only access the site that gives you the service and none other.

Leo: Yeah, yeah. Love it. Steve, one more, you know, I'm going to make one more announcement. Now, tomorrow is Internet Freedom Day, another big protest on the Internet to preserve Net Neutrality. And you can find out more at BattlefortheNet.com. In fact, I encourage you, go there.

Steve: Yay, yes.

Leo: And send a letter to the FCC. Call your Congress Critter. A lot of companies participating in this, including, ironically, AT&T. Well, we don't buy that, AT&T. We know you're not a part of the fun. And a lot of websites you'll know will either go dark or in some way indicate it. Ours will with banners. And as we did with the SOPA protest some years ago, if you tune in tomorrow, any of our streams, they'll be in black and white.

Steve: Nice, nice.

Leo: With a badge just to remind you to call your Congressperson and email or write

to the FCC and comment. You know, the second stage of commenting I think begins tomorrow. They didn't get all the comments they ought to have because...

Steve: God help them.

Leo: ...they were DDoSed somehow by all the comments. You think they could pull that twice? I don't know. Anyway, so tomorrow's the big day, July 12th. And I know you care about that, and we all care about preserving Net Neutrality. We would not be here if it weren't for the fact that anybody can participating in the Internet as an equal.

Steve: And corporate dominance is just - it's a problem. We do need some regulation.

Leo: Yeah. And that's what we're saying. Don't regulate the Internet, just regulate some ISPs that would like to regulate the Internet themselves.

Steve: Right.

Leo: Unilaterally. Steve Gibson's at GRC.com. That's where you go to get SpinRite, for 39 years the world's best hard drive maintenance and recovery utility. Wow. And of course while you're there you could check out ShieldsUP! and all the other freebies he offers because he makes a living on SpinRite. Including this show. The audio's there, and transcripts, really good transcripts. Elaine Farris does those all by hand, listening, typing, at GRC.com. You can also get audio and video versions of this show at our site, TWiT.tv/sn. Best thing to do, if you ask me, would be to subscribe so you get every episode. You want the complete set, by all means.

You can even watch us do it live, every Tuesday, 1:30 Pacific, 4:30 Eastern, 20:30 UTC at TWiT.tv/live, or live.twit.tv. I think both work. I think even twitlive.tv or .com works. And if you do that, do join us in the chatroom. Nice people in there, including Dave Redekop. If you can just go to irc.twit.tv, you can participate, participate with them. What else? Anything else I need to say?

Steve: We're approaching Black Hat. I have a feeling we'll be discussing more pre-Black Hat news next week, and we'll be covering in detail as it happens.

Leo: It's always our busy time.

Steve: We've got to, yeah, I think we're going to have a lot of fun in the coming weeks.

Leo: Steve Gibson, thank you so much. And we will see you next time on Security Now!.

Steve: Thanks, my friend.