SECURITY NOW!

Transcript of Episode #513

Listener Feedback #215

Description: Leo and I discuss the week's major security events and discuss questions and comments from listeners of previous episodes. We tie up loose ends, explore a wide range of topics that are too small to fill their own episode, clarify any confusion from previous installments, and present real world application notes for any of the security technologies and issues we have previously discussed.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-513.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-513-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Lots of security news. Steve will have that and 10 questions and 10 answers. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 513, recorded Tuesday, June 23rd, 2015: Your questions, Steve's answers, #215.

It's time for Security Now!, the show that protects you and your loved ones online, with the Explainer in Chief here, Mr. Steve Gibson. Hi, Steve.

Steve Gibson: Hey, Leo. Great to be with you again. The last time for a month, apparently. You are...

Leo: I'm going away.

Steve: ...off on a river somewhere.

Leo: Yeah. And the good lord willing, the creeks don't rise, I'll be back on...

Steve: Oh, we're going to get you back.

Leo: I'm going to be back on July 14th, but I'll miss this show. I'll probably take the

15th off and come back that Saturday.

Steve: Yeah, well, you've got to have your time zones will be all messed up and upside down.

Leo: I'll be all wopperjawed, yeah, yeah. Hey, some interesting stuff. This show of all the shows is one that the catalog is important. People go back, and they go all the way back, 512 episodes, and they listen to every episode. As you remember, in the early days we stored our episodes for this show and a few of the earliest shows on America Online.

Steve: Yes.

Leo: An early bandwidth partner. Cachefly took it over for every show. But we had some older shows still on AOL. I know you knew about this issue. And I think we had copied everything over to CacheFly; but the links on our old site, and I think some of your links at the time, still pointed to the AOL versions.

Steve: Yeah. I have redirection a la backend software on GRC's server. And so when I saw that there was a problem - and you know our listeners pretty quickly started tweeting, saying, hey, you know, these links are all broken.

Leo: Where did they go? Which is amazing because they were all old, old shows.

Steve: Oh, I mean, and they even knew. It was like from 414 on. It's like, okay, thank you for doing the work for me. And so sure enough, I looked at the source, and I was making a decision at 414 where to send it. If it was earlier than that, I bounced it through Podtrac with a tail on AOL so that it would go get counted, then go to AOL. And if it was later than that, it would go through Podtrac and then pull from Cachefly. So all I had to do was remove a little bit of that code, remove the AOL stuff, after I verified that Cachefly had the entire library, as you said, as it does. And then GRC's links were okay.

Leo: Well, we've done the same thing, finally. We had the new website go up last week. And one of the nice features of the new website is we have an API that we can use. And Patrick Delahanty just used the API this morning to change all of those URLs on the website and in our database, which is managed by Drupal, to Cachefly. So everything should be okay. But, and you have the same problem, there were four weirdly numbered shows that we did in the 512. For some reason we foolishly, I foolishly decided not to just always do sequential numbers. At one point there are some letters in there.

Steve: Actually, there's a 215a, for example. I have exceptions in the registry. And so one of the things my code is check to see whether it's one of the wacky ones, and then it goes somewhere else in order to get that. But it, like, fixes it on the spot.

Leo: You're smart. You're smart. It turns out that all but, I think, two we were able to find. But there are, I think, one or two short episodes, I'm not sure even what they are, that are gone. They're not on AOL. They're not on Cachefly. You don't have them. There's a gap for you, as well.

Steve: If you'd tell me what they are, I do, because I actually have my entire separate archive of everything.

Leo: Yeah. Well, good. I'll tell you what they are, and you can let me know if you do because that would be great, and we'll make it complete. And somebody else will do it. I'll ask Patrick to fill me in on it.

Steve: There is something weird going on. And maybe you guys have just fixed it. But as of last evening, I've been getting some complaints that people are unable to right-click on your links, on the new links, and do like a save file or save link contents, whatever.

Leo: It's funny that they complain to you about that.

Steve: Well, it's because it's my podcast, and so it's like, oh, I can't get your - and it's like, in fact, it was when I was going through the mailbag.

Leo: It's an old bug. Just so you know, we fixed the bug. It's an old bug.

Steve: Okay. But it was still there yesterday.

Leo: Yeah. But, oh, you're using Firefox, aren't you.

Steve: I am.

Leo: So this is a weird Firefox bug. I'll pick a show. So what should happen, and happens with all browsers but Firefox, and we can't quite figure out why, you click Download Options, you'll get - and the reason we do this in a side-loading thing is because different options for different shows.

Steve: Oh, very nice.

Leo: So not every show has video, for instance.

Steve: Very nice.

Leo: And on Chrome, now, if you look at the URL, and I'll show you the URL, I'll just paste it in here...

Steve: It actually downloads a little piece of HTML which has a .MP4 extension.

Leo: That's Firefox misinterpreting it entirely. And I don't know why it does. If you look at the <href>, it is, it's the proper file type. And on Chrome we have a little - there's an HTML5 extension, the download extension, that means it will just go straight to a download on Chrome, which is what it should do. On some browsers, like Safari, you have to right-click and Save As. And on Firefox, right-click and Save As looks like it thinks it's saving an MP4, but in fact saves a 200-byte, as you say, HTML file. And we just...

Steve: Yeah. So for what it's worth, if anyone can hear this, and they want to use Firefox, you just rename that little file html. Then you click on it, and that brings up...

Leo: Oh, that's funny.

Steve: And it works. It brings up the little page that says "object moved to," and then there's a link. That one you can right-click and then Save As.

Leo: Yeah.

Steve: So, and you're right, people have said they're able to stream it, but what they want to do is save the file itself.

Leo: Well, if you start the stream, by the way, you can right-click on the stream and save as, and that works fine, too.

Steve: Right.

Leo: We're a little puzzled by Firefox's behavior there. It should be getting just a plain link to an MP4, and for some reason it's confused.

Steve: But I would consider that you've had, like, minimal problems, given the immensity of what you've done. Basically, you've created a major website and brought it up in, like from nothing, in no time. So, yeah, it's typical.

Leo: Yeah. Thank you. Yeah, there's usually a few bugs. That download bug was really an error on my part, and so but our team at Four Kitchens quickly fixed that, like in a half day. But then we still can't figure out what the deal is with Firefox. And

I just don't know. And this show particularly has a lot of Firefox users listening to it.

Steve: Right.

Leo: Chrome is by far the biggest browser. It works fine with Chrome, Safari, Internet Explorer, Opera. It's Firefox. I don't know why. And I guess, I'm told, Firefox 40, if you're on the beta channel, that's a future Firefox, somebody said it works fine. So...

Steve: Oh, good. Yay. Well, so it'll get itself fixed.

Leo: Maybe they're fixing it. I don't know. But Patrick Delahanty's in the chatroom, and he says that the Four Kitchens people are working on it and trying to understand it.

Steve: Cool.

Leo: So I don't know exactly.

Steve: So we've had a big week. Two major companies have major problems. One is what I would call a significant cross-application security flaw, which is Apple's, which affects both Mac OS X and iOS. We're going to talk about that in some detail, although, as you'll see, I intend to cover it more next week because I didn't budget enough time, and this thing is so comprehensive that I want to make sure that I understand it. It's not as simple as, like, oh, whoops, they have a buffer overrun or something. It looks like a major architectural problem, which sort of explains Apple's problems with getting it fixed.

Then of course we've got the problem with the Swift keyboard, which is installed by default on Samsung phones and affects about 600 million phones, which has generated a lot of news. Someone, a security researcher, did some LastPass exploit math in the real world, which gives us some numbers and further should make everybody feel comfortable about the virtual unhackability of their password based on the leakage that may have occurred from LastPass's network that we discussed at length last week. Some interesting news, really encouraging news about the future of web applications. Let's Encrypt has a launch date. We've got a few media updates. And this is a Q&A, so it's our 215th Q&A. So lots of great feedback from our listeners.

Okay. And I always do a Picture of the Week. This one on the first page of the show notes shows the oclHash - oh, I wrote "olc." I thought it was "ocl." Anyway, the Hashcat, which is the GPU-based, sort of almost hardware-level, not ASIC speed, but GPU speed, doing 100,000 iterations of the password-based key derivation function for SHA-256. And so here's a GPU cranking at speed, and it is delivering 2,577 guesses per second. I quoted, I think it was 8 or 9K guesses per second from an ASIC-based system. So these numbers are coming out about right. And based on this, as we'll see, Robert Graham, who is a longtime security researcher, he was the guy behind - remember, Leo, back in the day, the BlackICE firewall.

Leo: Oh, yeah. Loved that.

Steve: Yeah, BlackICE was Robert's company. And so he was curious. And so he ran the numbers to give us a sense for what it will take to crack LastPass, given their 100,000 iterations. And the upshot of his note is that that's so many iterations that, as I said last week, they're taking the brunt of the execution even by offering that kind of security.

Leo: Yeah, no kidding.

Steve: Because, you know, every time a user does something, they've got to crank through 100,000 iterations in order to get to the other side of the pre-iterated password blob that the user sent them in the first place.

Leo: So that happens every time I want a password from LastPass?

Steve: No, no, no. That's any time you need to access the cloud in order to synchronize between devices. But still, that's...

Leo: Well, that happens frequently; right?

Steve: Yeah, it does happen frequently.

Leo: Wow.

Steve: And so they're doing some serious crunching on our behalf.

Leo: Thank you.

Steve: But the return, yes, the return for that is that the bad guys all have to do it. And they're doing...

Leo: And we're doing our own on our own end.

Steve: Right.

Leo: As you know, if you listened last week.

Steve: Right. And thanks to the salt, which is per user, there is no way anyone can do this once for all people. So that just, you know, it really means that what may have been lost - and I stress that because we still will never know what escaped. They're just exercising an abundance of caution.

Leo: The good news is the word got out. My mom, who I moved to LastPass last year, asked me...

Steve: Yes, Jenny said to me, "Is LastPass dead?" I said, "No, honey, it's fine."

Leo: My mom said, "Should I change my master password?" And I know what her master password is because I set it up. I said no. Just leave it. You're fine.

Steve: That's exactly the right response.

Leo: Yeah.

Steve: So the big story, and this is going to - I don't know what the future holds for this. Some research was done by two groups, security researchers in universities, on what they call sort of the - they portray it as a generic question, that is, how well separated are applications from each other, that is, how well is application isolation implemented in the real world? And they explain at the beginning of their paper, well, in fact, I'll just share the top of this because it sets it well. They said: "On modern operating systems, applications under the same user are separated from each other for the purpose of protecting them against malware and compromised programs."

For example, we talk about how you should not run as an admin, meaning that when you're a standard user, your privileges on your own operating system, you the user, who then runs applications on your behalf, those applications have your privileges. And those reduced privileges keep even a bad application from being able to wreak as much havoc as it would otherwise. So this is containment which is now sort of a given across all contemporary operating systems.

So they say: "Given the complexity of today's OSes, less clear is whether such isolation is effective against different kinds of cross-app resource access attacks," and they turned that into an acronym. So cross is X, application A, resource R, attacks A, so XARA, X-A-R-A. "To better understand the problem on the less-studied Apple platforms, we conducted a systematic security analysis on Mac OS X and iOS." I think it was 10.10.3. "Our research leads to the discovery of a series of high-impact security weaknesses which enable a sandboxed malicious app" - that is, a malicious app that should be sandboxed - "approved by the Apple stores" - and they said "stores" because it's both Mac OS X and iOS - "to gain unauthorized access to other apps' sensitive data.

"More specifically, we found that the inter-app interaction services, including the Keychain, WebSocket, and NSConnection on OS X and URL Scheme on the Mac OS and iOS, can all be exploited by malware to steal such confidential information as the passwords for iCloud" - that is, other applications' passwords stored in iCloud on their behalf - "email and banking applications, and the secret token of Evernote, for example." They said: "Further, the design of the application sandbox on OS X was found to be vulnerable, exposing an app's private directory to the sandboxed malware that hijacks its Apple Bundle ID." The Bundle ID is that unique token - every application that Apple has in their Apple stores has a token, and Apple guarantees its uniqueness. And so it's that unique ID which is used sort of to create a branch off the master root directory and so forth, in order to give each application its own sandboxed space.

They say: "As a result, sensitive user data like the notes and user contacts under Evernote and photos under WeeChat have all been disclosed. Fundamentally, these problems are caused by the lack of app-to-app and app-to-OS authentications. To better understand their impacts, we developed a scanner that automatically analyzes the binaries of Mac OS X and iOS apps to determine whether proper protection is missing in their code. Running it on hundreds of binaries, we confirmed the pervasiveness of the weaknesses among high-impact Apple applications. Since the issues may not be easily fixed, we built a simple program that detects exploit attempts on OS X, helping protect vulnerable apps before the problems can be fully addressed."

So, okay. So that's how their 23-page PDF begins. And I started to plow into it, and I realized, okay, wait a minute, I'm not going to be able to really get my brain around this in time for the podcast. So what we know, what is commonly known, and I hope to nail this down further, mostly because now I'm really interested in understanding the nature of this, is that a malicious application has the ability to obtain unauthorized access to other apps' sensitive data such as their passwords and tokens which they're storing in the cloud, in mail, and, for example, the web passwords. They demonstrated access to all of the web passwords stored by Google Chrome.

Now, Google was very responsive. The Chromium team immediately yanked iCloud support from Chrome months ago. Apple has been sort of oddly unresponsive. They were notified back in October of 2012, and at the time asked for six months of silence. Then four months later, in February of this year, 2015, their security people asked the researchers for an advance copy of the paper and for another six months. Now, at first I was annoyed with Apple, thinking, okay, come on, get going. Aren't you taking this seriously?

But again, for what little understanding I have, I don't have the kind of real grip on this that I want to, this is feeling more like a very significant, fundamental problem where Apple is maybe stuck in a position of not being able to make the changes they want to because it would break a lot of existing applications. And various blog postings, for example the 1Password people have posted, I think they're [AgileBits], have posted, saying this cannot be fixed on the app side. And there was somebody else on the application side. Oh, I think it was in the Chrome blog. They did what they could. But the Chromium guys said, you know, we've done what we can, but this can't be fixed without OS changes.

So from the 1,612 OS X and 200 iOS apps that this group scanned, 88.6% of them were found completely exposed, as they put it in their paper, to unauthorized cross-app resource access, that is, this XARA attack, which would allow malicious apps to steal otherwise secure data. They developed the technology to bypass Apple's app store security checks. Now, I don't know, it might be that Apple can strengthen their security checks specifically to look for malicious apps that would do this. That might be one of the things that's going on. But they were able to get their - even having told Apple in October of 2014, in January they got their malicious, I mean their benign but malicious simulating test apps approved by Apple in the App Store in January of this year.

So the feeling I've got is that, I mean, so there are several things. It appears to be possible to test for this being exploited. So that may be one of the - I would call that a "mitigation" that Apple may be working to employ if they're, like, stuck right now. The sense I get is that this may require a coordinated significant change to the OS and applications. And although it was a small sample, 88.6%, which is a very high percentage of apps, weren't, again, as I understand it vaguely at this point, there are authentication measures which can be employed. But Apple also never told app developers to do it. So it looks like there are some apps that did it, for whatever reason. But Apple at least was never clear for doing these things that the apps should implement stronger authentication measures. So most didn't.

So again, I'm sorry to be as vague as I am. I had to talk about it this week. I'll be fully tuned up for the next podcast. But it's a serious, serious piece of research, 26-page PDF of details. So I'm, as you can imagine, I'm really curious to get a better understanding of exactly what this means. And not being an Apple developer, it takes a little bit longer to come up to speed on terminology. But this looks like a not good thing. And Apple has basically been saying for, what, since last October, so, what, eight months, please don't tell anybody about it. Well, now it's out. And hopefully Apple is close to at least, for example, using the detection technology to keep bad things from happening. And if malware can get through their prescreening, then this represents a problem for Apple users.

Leo: And we mentioned this, of course, on MacBreak Weekly earlier. And iMore has been all over this. Nick Arnott there has a great analysis of the XARA, the four different XARA attacks. And Rene Ritchie has a human-readable synopsis. But they also asked Apple for a comment. And Apple told them, quote, "Earlier this week we implemented a server-side app security update that secures app data and blocks apps with sandbox configuration issues from the Mac App Store." So that's for OS X.

Steve: Good.

Leo: We have additional fixes in progress and are working with the researchers to investigate claims. Boy, you know, it does feel like that's something that should have happened in October.

Steve: Exactly. It's like, I mean, first they ask for six months, and then they waited until they only had two months left to ask the researchers for their research.

Leo: Yeah.

Steve: And then said, oh, my god, give us six more months.

Leo: No.

Steve: And the researchers are like, hey, you've already had eight. So, you know, we've got to publish this.

Leo: I think your scenario that the fix may be a big problem for some apps is probably accurate. I mean, I think probably what they're looking at is, well, who does this affect, and how?

Steve: Right, right. And so, like, there's the question of apps...

Leo: And a lot of apps rely on this URL scheme for interapp communication because Apple frankly failed to provide them with anything better.

Steve: Right. And as I understand it, one of the common things that can be done is the exploit occurs when an app, for example, Access, uses the Keychain in order to get or verify a password. So normally apps would be able to do that. They might have it cached or do it in a way that is not being seen by the user. But as I understand it, one of the ways the malicious app acts is to induce a vulnerable app to cause the user to be prompted. And through that mechanism, the act of a vulnerable app then using some IPC mechanism, an interprocess communication, the malware is able to catch it in the act, and the flow of information leaks.

And so, as I understand it, one of the signs is, if an app that normally isn't explicitly prompting you about the Keychain does so, that could be a flag that this is going on. And so again, that sort of also speaks to the idea that, wow, I'll bet there's a way to detect that. And that's what these guys have. They have a detector that can catch it happening. So it's not something that our systems are looking for now. But I'll bet you that that'll be something that gets put in.

So, yeah, it feels like it's like, whoops. Apple should have always been telling people to do much more to guard against this. And, you know, maybe - and it sounds like it's not something Apple can also easily fix. And of course we already have tons of apps installed. And so whereas Apple can say we're going to go through the app stores and check for apps and block apps that don't have this fixed, I mean, so that will be the mechanism where there's the pushback to developers to add all this extra authentication, which they just, you know, Apple apparently didn't tell them they had to add. But some did, but way most didn't. So, yikes.

Leo: Interesting, yeah.

Steve: Okay. The Samsung problem. Samsung phones have a keyboard installed, so-called "factory installed." It's the default keyboard from Swift. And of course the SwiftKey people have been very quick to say, whoa, wait a minute, this is just some of our core technology was licensed. This is not the SwiftKey that you install. This is the keyboard that's already there, and it happens that Samsung got that from us, but we really don't have any responsibility for it, so don't blame us for this.

There was a lot of concern about this, and it's potentially warranted. But I'll explain the nature of the problem and why and what it takes to exploit it, and so it's somewhat less of a concern, or at least we can calibrate the concern. So this affects at least, like we know, more than 600 million devices were initially vulnerable. I created a bit.ly link for a page by the guys that found it, the Now Secure people. They've got a page where they're monitoring known patches. Everything is vulnerable initially, and they've either got some still vulnerable or some unknowns, nothing yet shown patched.

But bit.ly/sn-sam, S-A-M. That was the shortest one I could find that worked. And that will bring you to, if you go down about halfway down that page, bit.ly/sn-sam. So drag the scroll bar about halfway down, Leo, and you'll see, there it is, a list of Samsung phone models and carriers, because this is a per-carrier thing, and the status of whether this has been fixed yet or not as far as we know. And I wanted to give people the link so they could check it from time to time.

Leo: I can simplify it.

Steve: Yeah, I know.

Leo: Nobody's fixed it. It's either not fixed or unknown. All the Galaxy S6's from the main carriers, all the Galaxy S5's from AT&T, Sprint, and T-Mobile, all the Galaxy S4's from everybody, all the Galaxy S4 Minis from everybody. So basically, if you've got a Galaxy S4 Mini, S4, S5, or S6 from any of the big U.S. carriers, you're at risk. And I do, which is why I'm interested in this.

Steve: And so I'll explain exactly what the risk is. The problem, that is, what can happen is that, if this keyboard software is exploited, and I'll explain in a second the vulnerability and the exploit, an attacker could remotely access sensors and resources like GPS, the camera, and the microphone; install malicious applications without the user knowing; tamper with how apps work or how the phone works; eavesdrop on incoming and outgoing messages or voice calls; and attempt to access, probably successfully, sensitive personal data like pictures and text messages.

So this is really bad. I mean, this is like a rootkit, essentially, on your phone, which can be installed, unfortunately, without the user doing anything explicit. But it's not like it's an instant 100% takeover, as I'll explain. So this is all thanks to a security researcher, Ryan Welton at NowSecure. Samsung was notified last December 2014, so last December, or December of last year. So seven months ago. Samsung has said that they - and Google and the Android project fixed this some time ago. The problem, of course, is carriers. Samsung has provided patches, they've said, to some carriers. And some carriers are said to have updated, but we have no confirmation of that at this point.

The one thing I wish we had and we don't have is a test. What we really need is a test. And it turns out that it's not even easy to use the version number of the keyboard that's built-in because that's not reliable. So, and the reason we don't have a test, like a site you could go to or something you could do, is also the good news because it's not that easy to do. This all happened a week ago while we were doing this podcast, talking about the LastPass hack. The Black Hat conference in London was underway, which was where Ryan disclosed this. Proof-of-concept code is up now on GitHub. So everything that bad guys need in order to do this is there.

There's a video on YouTube showing this happen, which didn't make a lot of sense to me until I went through Ryan's blog posting, which explains what the things are, like explains what's going on. Then, when you look at the YouTube video that they made, it's like, oh. Now I know why he highlighted that line. So in proper order, if you're curious, go to the NowSecure site, find Ryan's blog. I don't know if I have a link here to his - I have a link to the YouTube video. I don't think I have a link to his blog posting. But you want to find that, where he takes you through it. You can also find his slides from his presentation, as well.

[https://www.nowsecure.com/blog/2015/06/16/remote-code-execution-as-system-user-on-samsung-phones/]

Okay. So what was written is that a remote attacker capable of controlling a user's network traffic can manipulate the keyboard update mechanism on Samsung phones and execute code as a privileged system user. That's not quite root privilege. But system is next in line in power. If you have access to system-protected files, you still have all you need in order to do everything I said above. Basically, it's tantamount to being root.

"This can be exploited in a manner that requires no user interaction. A user does not have to explicitly choose to download a language pack update to be exploited. The Swift keyboard comes preinstalled on Samsung devices and cannot be disabled or uninstalled."

Leo: Yeah, see, that's the problem. I don't use it, but it doesn't matter. It's still there.

Steve: Right. It's still there. Even when it is not used as the default keyboard, it can still be exploited. Now, okay. Get ready, everybody. Sit down, because this is just so sad. The keyboard issues, when it wants to check for updates or when the phone is restarted, the keyboard issues an HTTP get command. And I did not say "HTTPS." It issues an insecure GET command for a zip file with a well-known nonchanging URL to SwiftKey.net/samsung/download/ and then some other stuff dot zip. That file is the keyboard update file, which is transferred in the clear, in plaintext.

And so the first thing that Ryan did was he - oh, and so what this means is that anything that can change the result of the phone's fetch of an HTTP plaintext file can implement this exploit. And so what that means is essentially a man in the middle. So the good news, it's not like somebody on an open WiFi who simply had access to the same network that you have - oops, sorry. Had to silence an iPad. Not like somebody on the same network could get you. You would have to do an ARP exploit. You have to do something to intercept. And in Ryan's demo he's running a little MITM proxy. So he's intercepted his phone's Internet access and catches the phone issuing this http://get command. And he replaces the zip file that it would download with his own. Now, the first time he tries it, it doesn't work because it turns out that there's a hash, an SHA-1 hash, which has to match the hash of the zip. But the SHA-1 hash is also obtained over HTTP.

Leo: We'll just give you a different hash. What the heck.

Steve: That's right. We're going to protect this with a hash, but we're not going to protect the acquisition of the hash. So he said, okay, fine. So then he created his test exploit payload, did an SHA-1 of it, then first changed the hash in the phone, then gave the phone his exploit payload. And through some other machinations, basically the files that are normally downloaded are just JSON small databases and language files...

Leo: They're text files, yeah.

Steve: Yeah, they're nonexecutable. But since the keyboard itself, the keyboard process is running with next as good as root privileges, system privileges, it turns out it's very simple to get code to execute, and he did that. So the situation we have - oh. And every one of these packages is version-dependent. It's device-specific. But the query that the phone makes has the standard query headers, including the user-agent and all of the details in the user-agent to pick from a library of pre-prepared exploit packages. So the fact that they're, like, per phone doesn't matter. And you were running the YouTube video there. It's like a three- or four-minute video that, you know, demonstrates this happening.

So bottom line, what this means is that until essentially all of these affected Samsung Galaxy S4 phones from 4 Mini up through the 6S or the S6 that just happened, until they're updated, they're vulnerable to a not difficult, but also not easy to implement, man-in-the-middle attack. Oh, and note that this only happens if, while there's a man-in-the-middle installed, that is, somebody who's managed to intercept network traffic in the line somehow, which requires some effort, the phone's keyboard, the default-installed keyboard, has to make this query in order to get the malicious payload in return, which it only does periodically when it just chooses. It's like, well, I haven't been restarted for a long time, so I'm going to check for updates. So when it does an update check, or when you restart the phone. So a phone reboot, when the phone then acquires a network connection where someone maliciously is intercepting it, which again is not trivial to do, that's where the problem is. So it's not really bad. But if someone can exploit it, then that's really bad.

Leo: That would be bad, yeah.

Steve: Yeah. So it's...

Leo: As somebody in the chatroom pointed out, if you didn't use WiFi, if you just used your LTE, you wouldn't be vulnerable.

Steve: Correct.

Leo: Somebody has to get a man in the middle on.

Steve: Good point.

Leo: Yeah.

Steve: Yeah, that would require a much more, you know, NSA-level sort of exploit, where they have access to the Internet, and upstream of your cellular carrier they would be able to say, oh, look, this is the phone we've been wanting to install some malware into. So a state actor could do it over cellular. But you're right, no random hacker in an open WiFi network could do it. Certainly they could, if they could get your phone to restart or, like, maybe do some social engineering, say hey, let me show you something, like restart your phone and then blah blah blah.

Leo: Get on my WiFi network named FBI Surveillance Fan. The fix would be fairly simple. Just turning those into secure links would solve that problem. You'd have to have a secure server, obviously.

Steve: Yeah, exactly. Turn those into secure links.

Leo: But they have to get it onto the phone. That's the thing. It's easy to fix. But you've got to get Verizon to push it.

Steve: That's the problem. As we've seen, there's a problem with older versions of these phones just never getting pushed. And Samsung seemed a little unclear about, like they said we've provided some patches to some carriers. Like, okay, this is bad, you know, fix this. So we'll have to keep an eye on it and see what that page updates as it goes. These guys, the NowSecure people, are on top of this, and this was a big find and win for them. Their business is mobile security, so this is right in their wheelhouse. And I'm glad they'll stay on top of it and let people know as phones get fixed. And it's the kind of thing you want to maybe stroll into your local carrier and say, hey, you know, have you heard about this bad problem? When are you guys going to get it fixed? And put some pressure on the system in order to make this happen because this is potentially not good.

Leo: And to reiterate, this is not a problem with the SwiftKey keyboard you download.

Steve: Correct.

Leo: This is only a problem with those specific models of Samsung Galaxy phones. They come with a Samsung-licensed version of SwiftKey that's flawed.

Steve: Correct.

Leo: And unfortunately, you can't uninstall it. I wonder if you can disable it, and I wonder if that would make the difference?

Steve: No. They said disabling it, neither disabling nor uninstalling it is possible. It's still...

Leo: Yeah, people are going, is - no, no, it's just these versions of the Galaxy phones. And there's nothing you can do about it because it's installed.

Steve: Yeah. And I just wish there was a test because that would be great, if there was like a benign test. Then people could check whether their phone was still vulnerable or not. So maybe somebody within range of our voice will be able to do a test. That would be great. If so, let me know.

So Robert Graham of BlackICE firewall fame, one of the early application-based firewalls, has ErrataSec.com, which is, you know, we've referred to ErrataSec and Robert Graham over the years. He's active in the security space. And so he was just - he just did a blog posting last week with the title, "Should I Panic Because LastPass Was Hacked?" And so the very first, the top of his statement says: "Maybe not. LastPass uses 100,000 iterations in its PBKDF2 algorithm. If you choose a long, non-dictionary password, nobody can crack it. Conversely, if you haven't, then, yes, you need to change it."

And as I mentioned at the top of the show, his GPU-based Hashcat system was able to run a 100,000-iteration SHA-2-based PBKDF2 algorithm, which is what is being used at the server farm at LastPass, at the rate of 2,577 hashes, which would be password guesses per second. So Robert's math, he attempts to do the math; but unfortunately he hasn't got his alphabet right, unfortunately. But what it means is that it's actually much stronger than he says.

He says: "Consider normal hashes, not the stronger ones used by LastPass. My desktop can crack one billion of those per second." So that gives you also a sense of scale. A billion hashes per second and 100,000 iterations of PBKDF2 reduces that billion to less than 2,600 hashes per second. So huge scale down in terms of cracking. And he says: "Consider that a password can be built from upper and lowercase letters, numbers, and punctuation marks, or about 64 variations per character."

Well, when I saw that, I said, wait, no, I'm sorry, it's 95. I know it because I did Password Haystacks, and I've counted them all. There's 26 lowercase, 26 uppercase. So now we're at 52. There's the digits, so now we're at 62. And there's 33 punctuation characters, bringing us to 95. So his math is based on an alphabet of 64. In fact, it's another third bigger, which is significant. Actually, no, it's another half bigger. It's half again.

So he says, in this case, a five-letter password has a billion combinations, so a fast computer can guess it in a second. Adding one letter with its 64 different possibilities - and again, he's multiplying by 64. I would argue you'd multiply by 95, and you'd already be up at a lot higher number. But that makes it 64 times harder, meaning it'll take a minute. Another letter, now we're at seven letters, and it becomes an hour. Another letter, to eight, becomes several days. Another letter, now we have nine digits, and it becomes a year. Another letter, 10, and it becomes 64 years. Another letter, 11, and it's thousands of years. Another letter, 12, and it's millions of years.

So he says: "LastPass rehashes the password 100,000 times, which slows this down dramatically. What I could have hashed in an hour now takes a decade. On the other hand, consider an adversary like the NSA or a hacker with a botnet that controls 100,000 computers. That would speed things up back to the normal rate. But even with 100,000 computers, the NSA won't be able to brute force a 12-letter random password.

"Unfortunately," he writes, "brute force isn't the only option. Hackers may instead use a dictionary attack, where they use word lists and common password choices, like GoBroncos!," he writes, "and then mutate them with common patterns like adding numbers onto the end. This speeds things up dramatically, making it easy to crack even 12-letter passwords in minutes. In between the two are Markov chains."

And that's something we've never talked about on this podcast, but it's certainly a powerful tool. A Markov chain is a probability network where you're at a given node, and there's a certain probability, for example, you're at a node that says lowercase alpha. So what's the probability of the next character being another lowercase alpha, which would be a loop pointing back to that node with a certain probability, or a lowercase alpha being followed by a number, which would be an arrow going off to the number node with a certain probability. I actually started the work on, was it Haystacks? It was either Haystacks or Off The Grid, using a Markov-based password modeler because it's such a powerful technique.

But Robert notes that, in between, that hackers are now using these because Markov chains are sort of like brute forcing, but they follow more of a human pattern, like sort of modeled the way humans tend to construct passwords. So they don't bother with super high entropy first. They try lower entropy solutions. Anyway, so he concludes, then, "The upshot is that your 12-character password is a lot weaker than" - oh, of Markov chains - "is that your 12-character password is a lot weaker than you assume. So your passwords not only have to be long, but also fairly random and not based much on dictionary words, and random in ways that Markov chains can't easily guess."

So what I would recommend our listeners do, if they're curious - because my math is correct over on the Haystacks page [grc.com/haystack.htm]. Now, we know that a GPU can crack at about 2,600, a little slower than 2,600 guesses per second, and that an ASIC-based system, much more pure hardware specialty, can run about 9K. Given 9,000 guesses per second, the Haystacks page will show you, securely, I mean, don't put your password in, but put one in that's sort of like the passwords you have, because that's all my page cares about. It characterizes the number of different types of characters you use and shows you that resulting exact number of combinations there are for that kind of password and at its length. And then consider maybe 10,000 guesses per second, and you'll get a good sense for how secure your password is against one adversary with a hardware ASIC cranking away.

Now, of course, we've got server farms, thanks to Bitcoin, of this kind of superfast SHA-256 hashing. But again, they only scale up by the number of units that you have doing the work, which is not typically hundreds of thousands. It's maybe 10s or 20s or 100s. So anyway, it does give us sort of a benchmark that we can use to get a sense for the length of our passwords and how much full of junk they are, unpredictable stuff.

And really cool news. We've talked on the podcast forever about various aspects of code running in browsers, JavaScript and its benefits, its issues, and all the ways that the various entities, like Mozilla with asm.js, which we'll remember is the approach Mozilla took was to come up with - to recognize that the problem with compiling JavaScript is that it is a so-called "dynamic language," where variables don't have to be defined. They can even change the type of contents, whether they're a string or a numeric or a floating value, at will. I mean, it's the kind of things that drive compilers crazy, to track all this. And they also have something known as dynamic memory allocation and garbage collection, where you need a so-called "garbage collector" to decide that you're no longer going to use this variable, and you can release its memory. It's really a nightmare for compilers. So what the Mozilla guys realized is, hey, we could still have JavaScript, but just not use a lot of these crazy features. And if we do that, if we come up with a subset of JavaScript, that would be much more compiler friendly. So that's been Mozilla's approach.

Google has a system that we've talked about called "Native Client." In fact, we referred to it a while ago because there was an exploit against it, and I was looking at how much Google had had to go through in order to make Native Client safe. Native Client is actually x86 or ARM machine language which is essentially sandboxed, which is difficult to do. And I was impressed by what they went through to do it. But that's been their approach. Then they did something called - and that was called NaCl for Native Client.

Then they did something called Portable Native Client (PNaCl), which is actually a bytecode, where instead of compiling to x86 or ARM machine language to run on either of those platforms, they compile to a bytecode and then have an interpreter. And of course bytecode is a time-honored solution for getting machine independence. That is famously what Java does. Java has the so-called JVM, Java Virtual Machine. And the virtual machine-ness is that it pretends to be a computer that natively executes bytecode, except no computer natively executes bytecode. Instead, this JVM is a thin layer of translation that interfaces the standard bytecode to the specific processor that you're running on. But that's how you get a Java system is able to run on a Mac in the old days when it was on the PowerPC platform, or on a PC, or on an ARM platform, or anything else.

And of course Microsoft has been very involved with bytecode. The whole .NET system is a bytecode system where they have various high-end language compilers that all compile down into what's called the Common Language Runtime, the CLR. And that is bytecode. So that is a runtime that all their various languages compile to. Then they can invest heavily in a single virtual machine to execute that.

So here's, I mean, I just couldn't believe this when I saw this. Google, Apple, Microsoft, and Mozilla are all partnering. No more this is how we're going to do it; this is how you're going to do it; we're not going to talk to each other. It's on GitHub. It's called WebAssembly. It is going to be a single universal bytecode for the Internet, for web browsers. So what it will mean is that, rather than web browsers downloading an ASCII JavaScript and then having to go through just-in-time compilation and all that, I mean, certainly there are some benefits to the people who are providing this because what will happen is, now we will download binary. The browser will get a binary blob which has already been compiled into bytecode and be able to execute it immediately. It means it'll be way smaller, so download times get faster. It also, because it's compiled, although it can be decompiled, of course, it does provide a great deal more intellectual property protection for those who are wanting essentially to allow users of their code to run their proprietary code in the user's browser.

It's not perfect. Again, as we know, it's not - you can't protect against that kind of reverse engineering. But it does mean the system can be very fast. And in fact they're seeing a 20X speed improvement. They have a bunch of goals for making it portable, size and load time efficient, in a binary format, which will be directly downloadable and can be executed by a next generation of virtual machines that'll be common across the browsers. I'm sure the different implementations will work to make theirs faster. There'll always be competition. But there'll be this universally common agreement. We have it now at the JavaScript level, we've got common agreement there - or ECMAScript, as it's called officially. But this will be a layer below, like a much closer to the metal common language that is this bytecode.

And what's interesting is that the initial functionality is targeted at C and C++. So you'll be able to author code in C and C++. There will be compilers to compile that to this new WebAssembly. And then browsers will be able to run it. It interfaces with JavaScript. There'll be a middle layer to compile JavaScript into this bytecode, not as efficient, probably, as maybe starting at C and compiling into the bytecode, just, again, because of JavaScript's oddities. But still, they've worked out essentially a complete development path. Again, GitHub, and it's called WebAssembly. And they're on it. So I'm just jazzed. This looks like, you know, this sounds like really what we're going to see for Web whatever point-0 we're on. Are we on 2.0, and we're going to go to 3.0?

Leo: Oh, we're way beyond 2.0.

Steve: We're beyond 2.0.

Leo: Oh, that was years ago.

Steve: Okay. So whatever point-0 we are on, this will certainly count for one. And we have a date from the EFF and the Let's Encrypt project. The week of September 14th will be the launch date for the Let's Encrypt project. So it's still a ways away, a little bit later than we were expecting. We were hoping it for about midyear. But these things always take longer than you expect. But they are now committing to the week of September 14th, Let's Encrypt will come online. And as our listeners know, I'm excited about it for the, you know, it's not bell-shaped, it's not even a pyramid, sort of the distribution of domain-only certificates versus everything fancier, the domain-only certificates, where you're just needing to say I want SSL/TLS, and I want HTTPS connections, and all I really care about is that security, giving people a good feeling about being on the website because we are secure. And that's all we need is verifying that we control this web domain. That's what Let's Encrypt does.

I'm still going to have EV certs from DigiCert because I want green. I want more. But most people neither want nor need nor want to pay for more. And so Let's Encrypt is free, and it completely automates the entire process, basically allowing the server to autonomously negotiate with the Let's Encrypt facility in order to get it to issue itself certificates in a secure fashion. So I'm excited to see this happen. This will be a nice change for that huge majority of the 'Net that would like to move to security, but, I mean, very much like TWiT was, Leo, where it's like you were saying, you know, we really don't need it. There's nothing here that needs to be protected. But we're going to do it because we recognize that that's the way the world is going.

And a lot of the world has been grumbling because they would have to pay something. Now that all goes away. And even the configuration. You just issue a couple commands into a non-Windows server, I don't know what they're going to do for Windows, but a couple commands into a Linux or a UNIX box, or nginx, and it just does it for you. You just, bang, you're now running secure. So middle of September.

Oh, and this is - okay. So now, miscellaneous stuff. I follow Matt Green. Matthew Green, of course, is the cryptographer at Johns Hopkins, who has been very active with the TrueCrypt audit and many things that we've spoken of. This just came out - this was just random. This was on Father's Day. It was a tweet at 2:37 in the afternoon. And Leo, I'd heard you on one of your other podcasts talking about this. And so this really struck home. So Matt tweets, and this is, you know, super crypto guy, right, I mean, genius, professor, everything: "My wife bought me an iPod Nano for Father's Day, which is awesome. But OH MY GOD ITUNES," he had in all caps.

Leo: That's a recent tweet? His wife bought him a Nano?

Steve: Yeah. This was Sunday. It was his Father's Day present.

Leo: That's very funny.

Steve: Yeah. I just got a kick out of it. But OH MY GOD, ITUNES. Imagine.

Leo: iTunes. Matthew, you've never used iTunes, really?

Steve: Yeah, imagine. I know he's in the Mac camp because he has an iPhone as his device. And so he's done things, but probably never really looked much at iTunes. And it's like, yeah, I mean, it is, it is a nightmare. And I've heard you guys talking about, like, it's time for Apple just to scrap it and start over because my feeling is, once upon a time, it was Apple trying to make the whole thing transparent and easy when all anybody had was an iPod. But now it's just become so burdened down with all the other things that Apple is doing beyond being an iPod. So, I mean, with the phone and the pad and now the watch. And it's like, okay, let's start over.

I did want to make a quick note. I got email this morning from this neat guy, Oscar, who is doing the PDP-8 recreation kits. People from time to time ask me what these lights are blinking behind me. And those were a very expensive - multiple thousands of dollars, due to the nature of it - kit that I purchased and built and programmed with a blinky light program. This guy has a beautiful recreation of an earlier generation PDP-8. I mention that because whereas mine has two banks of lights, this was DEC beginning to do like a cost-reduced version.

So the PDP-8/E, which is what this is a copy of, was the least expensive minicomputer at the time. And they said, you know, we really don't need a separate little region which decodes which instruction is as you're single-stepping along. But it's fun to have it. And they were like - and they have, like, other registers which are not shown here. Here there's a dial, and you have to select which register in the machine you want to bring out on the panel. On the PDP-8, I don't remember if it was an "I" or an "L," but it's the one that this guy's recreated, they're all out there. They've got five banks of blinky lights and a whole region of instruction decoder. Anyway, I've talked about it before. It's based on the - I'm blanking now. It's on the little Linux-based gizmo.

Leo: Raspberry Pi?

Steve: Raspberry Pi, thank you, yes, based on the Raspberry Pi. So basically it is a front panel and lights that recreates the original PDP-8/I or /L, whichever it was. And inside is a Raspberry Pi. And, for example, the OS/8, the operating system, he's got on a little thumb drive. And so you just plug it in, and you can run OS/8, the original PDP-8 operating system. Of course I've got PDP-8 pages on my site.

Anyway, I wanted to let everybody know that what I just got today was a PDF of the order form. He's all on track. He's moving forward. He's expecting to be shipping in, like, next month, in July. Everything I have seen from this guy says, like, the guy is doing a topnotch job. These things come with a beautiful box as part of it. You have to cut a hole in the back of it somehow, because that's not done, to run your cables out the back.

But anyway, for what it's worth, I wanted just to mention it. Once I have them behind me running, you'll see them again, and there will still be time to get them. But he is taking orders now. And I have a bit.ly link that I created back in April when I first mentioned this: bit.ly/pdp8kit. Not surprisingly, pdp8kit. For what it's worth, I mean, this is the one. And I forgot to say it's only a couple hundred dollars. It's 300 for the fully built kit, or the fully built finished unit; or less than 200 for one where he provides you with everything, and you put the rest of it together yourself. So, I mean, it's a great deal and looks just fabulous. So bit.ly/pdp8kit, and you, too, can have lots of - actually even more blinky lights than I have behind me, behind you.

Also in miscellanea, my Twitter followers already know this. In fact, they're glad I've stopped tweeting about it. Thanks to other Twitter followers, I was pointed to a puzzle to consider. It's called Infinite Loop. And it is great. I recommended people to Blockwick before, which I have really enjoyed. This is completely different. It is free. I don't know why it's free, but it is. It's available both for iOS and Android. I've got links in my show notes, or you can go to LoopGame.co, L-O-O-P-G-A-M-E dot C-O. And this thing has my absolute top recommendation. There's no timers. There's nothing bugging you, like checking, testing you to finish or anything. It's just really pleasant. I know that Leo and Lisa both liked Hoop, and we talked about - or, I'm sorry, Hook.

Leo: Hook, yeah.

Steve: We've talked about Hook before. And in fact I learned about it because it was one of your picks on iPad Today, before it became iOS today. And I liked it a lot. But it only had 50 levels. This thing never runs out. And because I didn't believe that, I took it to level 200. And finally...

Leo: Did you stay up all night?

Steve: No, but I relax during the day. It was on Sunday. And so I tweeted at 150, and I tweeted at 173, and also at 200. Anyway, I cannot recommend it more highly. It is very easy. It is very pleasant. You get to produce beautiful symmetric things most, you know, often. LoopGame.co. Top recommendation on a great puzzle that's available for both iOS and Android.

Leo: Nice.

Steve: And I do want to thank our listeners who responded en masse to my call for The New Screen Savers topic ideas. I now have an outline just overflowing with them.

Leo: Good.

Steve: They've been coming in via Twitter. And when I checked the mailbag for today's Q&A, it was, wow, I mean, just like - it was either that or talking about advertising and tracking. Those were the two topics.

Leo: No, let's not do that.

Steve: We're not, no, that totally dominated the whole...

Leo: Oh, I see, it's just people talking about it. Yeah, yeah, yeah.

Steve: Right, right, right. So I just wanted to thank everybody for responding to my call. Now the burden is to sit down and produce a hundred little shortcut, you know, fun tips and tricks.

Leo: Thank you. Nice. Thank you, thank you.

Steve: And so I will get about doing that. Three science fiction notes in sci-fi media. First of all, Syfy's new program, "Dark Matter," which I panned having never seen, was not as bad as I thought, or as I feared, from the little preview snippet I saw. I described it as something like bad actors reading a bad script or something. And, I mean, it's not wonderful. It's Syfy, you know, S-Y-F-I, or is it F-Y?

Leo: S-Y-F-Y, yeah.

Steve: Yeah, I think it is, S-Y-F-Y. It's Canadian, low-budget. But it wasn't horrible. It wasn't bad. And another show that just premiered and is still airing its first episode, "Killjoys," also just happened. As I said, the premiere episode is still out, and it runs on Friday nights. And it's some legally sanctioned bounty hunters in the future, kinda gritty. Somebody said it was sort of like "Firefly." And it's like, well, I wish it were like "Firefly." But still, looked good.

And then the last tip is "Humans" is airing, the first episode airing on AMC this weekend. It had a great debut in the U.K. a couple weeks ago. IMDB has it at 8.2 with over 1,400 users reviewing. And I've heard nothing but good things about it. It looks like high-budget, very polished, so it's just called "Humans." And we in the states can get it starting on this weekend on AMC.

And I did note that we are also going to have the start of "Mr. Robot" that of course I told everybody about weeks ago, that you were able to find everywhere. Our listeners went gaga over it, and so did the IMDB audience. With more than 13,000 people reviewing, it's a 9.4. And I don't think I've ever seen that on IMDB. And "Mr. Robot" also begins officially here in another week. So I was asking for lots of sci-fi, and suddenly it all happened at once, so I'm delighted.

I also saw in my Twitter stream a tweet on the 18th. I couldn't have been awake for it. I must have woken up and then looked back in the stream because at 1:35 a.m., or maybe it was his time, Peter Butler tweeted from his iPhone: "Hoping to have a testimonial in a few hours." Then it says "#fingerscrossed. Being SpinRited, soon to be SpinRitten." And he did send, he had attached a picture to it. And looking at it, as I did, it looked like a classic example of SpinRite's, like, what everyone wants to see. It looks like it's about half done, and 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 green R's, where there were at least 10 regions, maybe more than 10 sectors, but 100% green recovery of whatever it was that he was SpinRiting.

Then at 8:04 a.m., what, about 5.5 hours later, he also tweeted: "Worked a treat." Which I think is a British phrase, as far as I know, so that sort of would explain the time zone, if that's what it was. He said, "It's a brilliant product you have. Managed to save my ass again. I messaged your GRC feedback page."

Then he wrote, I thought this was sort of interesting, too, he said, "Hi Steve. I'm a CCTV technician from" - oh, Ireland - "Kildare, Ireland. I tweeted you a SpinRite pic earlier. A customer rang me with an error on his CCTV DVR." So that's like a security camera, a closed-circuit TV security camera. "The DVR had recorded some important footage; but when he was trying to play it back, the DVR was crashing and freezing. The recorder appeared to have written to a corrupt partition and was unable to read it back. A serious dumping incident" - which he doesn't explain, but that's apparently what the camera caught - "had taken place which the CCTV would have recorded, so the customer was very anxious to get the footage.

"So I thought of SpinRite. I'm an avid listener of your Security Now! podcast for the past few years and thought I'd give it a go. I started up SpinRite on Level 2 for data recovery and left it overnight." That was a short night. He said, "This morning I returned to work. I rebooted the machine, and it started with no problem. It was SpinRitten. I was even able to boot into the DVR software, review and download the important footage, and give it back to the customer. Thanks to all at GRC for an amazing product." He says, again, "It's saved my ass multiple times now. Thanks to Steve and Leo for a great podcast, too. Makes my commute to work on a Wednesday morning very enjoyable. Many thanks, Peter." So, Pete, thanks for sharing.

Leo: Very nice. And now, my friends, it's time for questions.

Steve: Let's start with number two. We can skip number one.

Leo: You want to jump right to number two.

Steve: Yeah, there was no big content there.

Leo: All right. Let me skip ahead.

Steve: I sort of already covered it.

Leo: Since we've already done it.

Steve: Yup.

Leo: Question 2. Andrew McGlashan - oh, I have a correspondence going with Andrew.

Steve: Oh, cool.

Leo: From Melbourne, Australia. We've exchange GPG keys, so we can talk to one another. He wants to double-check your measurements.

Steve: Andrew.

Leo: Andrew. When you quoted the comparative query counts and download sizes with Firefox's tracking enabled and disabled, couldn't the differences have been due to the browser's own local caching? I mean, you did get those pages twice.

Steve: I did. And I thought that was - yes. Fortunately, I am a very cautious researcher.

Leo: He's not foolish. He knew.

Steve: And when you bring up the Netscape Network Monitor in the Tools window, there's an explicit checkbox for "Disable all caching." And not only did I make sure that was the case - and, boy, I mean, yes, Andrew, you're 100% right, with that turned on, like, nothing happens. But it's like, okay, well, you know, because the browser says you already got all this. Yeah, I'm fine.

Leo: I knew it.

Steve: Yeah. And but for what it's worth also, the researchers made a point of doing 10 iterations of fetches, flushing the cache each time, and averaging them out. So everybody who was doing this understood that in order to get accurate net measurements, you need to average out, you know, you can't even just take even a single fetch because that could just be anomalous. There could be some big blob in a buffer somewhere in a router that just causes one particular item to choke out of the hundreds that your browser is getting. So, yeah, we all did this very carefully. And so these numbers are real, for what it's worth. But thank you for making sure.

Leo: Yeah. Josh in Atlanta, Georgia has questions about Wikipedia going to HTTPS versus descriptive URLs. He says: I've read several news article about how Wikipedia is going to SSL on all its pages. Most of those articles list the key features of moving to all HTTPS as to avoid censorship by blocking the data or eavesdropping by scanning your information. The problem I see is that Wikipedia and other sites like it use descriptive URLs. So if I want to read about communism, and I go to wikipedia.org/wiki/communism, this URL link is recorded by my ISP, could also be easily blocked by my ISP. HTTPS does nothing to prevent that; right? So I'm just trying to understand the benefit of SSL on sites with URLs that contain unencrypted content.

Steve: You know, and I thought for a while, wondering what the source of this confusion is because we've seen this before. Josh is not alone in thinking that the URL is not protected. And I think the reason is we can see it.

Leo: We see it, yeah.

Steve: We see it. And that I think confuses people. We see it in the URL bar of our browser. And so sort of the assumption is that whatever the security is that's happening, it's with the browser's guts or the data's guts, but not that thing that we see. But actually the way TLS works, and SSL has always worked historically, and of course HTTPS built on top of those two protocols, is that the connection is established between the endpoints, and the secure tunnel is brought up, meaning that the encryption wrapper is brought up to protect the data before anything is sent. So then the browser's first query of https:// and whatever, and all of the query headers, and all of that other stuff, all go through the tunnel. So your ISP who's carrying that traffic for you, and anyone else along the line, only sees gibberish going by. They do know that you're connected to Wikipedia. That we're not obscuring.

And of course that's what the whole Tor concept is. Tor attempts to obscure where you're connecting to on the 'Net to give you anonymity that way. So it is the case that someone observing your traffic could see that, first of all, your system would be doing a DNS query, perhaps, if it didn't already know the IP of Wikipedia. Then it would be actually making a connection to one of Wikipedia's IPs. That could all be seen. But once that tunnel comes up, and it comes up immediately, before any traffic goes through, there's no way for anybody to see inside.

Leo: Well, good.

Steve: And again, I think it's just because we can see it that it's like, oh, look, I can see it, so maybe other people can.

Leo: Yeah, that makes sense. Yeah. From, wow, Slovenia, Liam wonders how we securely transmit private data to tech-unsavvy people: You've said countless times how email is insecure, yet we still continue to use it because it's all we've got. Recently I've stayed at a hotel abroad which required me to send a photo of my passport over email ahead of time. Now, I can already hear your sigh of disapproval, but what could I do? I sent it, unencrypted.

This got me thinking: With PKI-inspired solutions to do this, it's up to the recipient of sensitive information to provide a secure way of transmitting it, not the sender, who is the person who will actually get hurt should the information fall into the wrong hands. How can we, security-conscious individuals, protect our data when we are the senders? Is there an easily linkable service, like MiniLock, where we can point hotel receptionists, accountants, secretaries, and other possibly non-security-conscious recipients of our data, which would allow them to quickly create a public key, send it, and decrypt what we send? This is a common problem we have to face. Is there a solution to this? Love the show. I've listened to it every week for a year or so, which will make it all the more embarrassing if this has been answered recently and I missed it. Liam.

Steve: Yeah, no, this was a really great question. And I tried to come up with a solution. I mean, I was immediately thinking MiniLock. But of course that requires the other end to get involved. And this is the whole problem that Liam's talking about is that, if you want to send something securely to someone else, you typically get their public key, like at a minimum you get their public key, and you encrypt using the public key, knowing that only they, with their matching private key, which they never disclose, are able to decrypt it. You might want to do the same thing at your end, that is, if they have your public key, you would also encrypt with your private key. That way they get the assurance that it came from you, and you get the assurance that only they can see it, if you want to do it both ways.

But Liam's mostly talking about how do I send something encrypted, in an encrypted channel, get it to somebody who has, like, zero tech savvy? And the only thing I could think was maybe like to use Google as an intermediary, stick something in a folder in Google Drive and then somehow arrange to share it. But the problem is how do you arrange to share it, providing them a link, for example, over a nonsecure path? I'd be interested to know if anyone has a solution for this, sort of a recipient, the non-tech-savvy recipient receives something securely over an unsecure channel. I agree. I don't - we really don't have anything at this point that occurs to me.

Leo: There are ways, obviously, to do the encryption part. As you mentioned, Google Drive, Dropbox would do that.

Steve: Right.

Leo: You could also use Zip and a password. The problem is transmitting the key somehow securely. And what I would say is just call them up, use Zip, for instance, call them up, say...

Steve: I think you're right because that's super simple.

Leo: It's a separate channel, anyway. I mean, it's obviously less. But call them up and say, hey, look, it's my passport. So I've zipped it up into a zip file to make it smaller for you. And when you try to open it, it'll ask for a password. The password is, you know, landshark. And that would, I think, give you at least a modicum of protection.

Steve: Right. I think that's a great idea. Yup.

Leo: Yeah. Yeah. You could do that with Drop - well, it'd be actually easy, you know, Zip would be the easiest way to do it.

Steve: I think Zip is perfect. I mean, you could, if you wanted it, you could make it an executable zip. So you send the - although EXEs won't go through email anymore, so that's a problem. So, yeah, probably just zip it up. Because, you know, all of our OSes now are able to unzip natively.

Leo: Yeah. And it's the same problem Caesar had with the secret encoder key.

Steve: Oh, you don't mean with his salad.

Leo: That's a different Caesar. But, you know, Julius had that problem because he would send - so they'd send two messengers.

Steve: Yup.

Leo: But using a separate communication channel, like the telephone, which is hard.

Steve: Yes.

Leo: You'd have to get both the email and the telephone. And Zip is secure; right? The password?

Steve: Yeah, it is, yeah. The Zip password is well implemented.

Leo: Good. Question 5 comes to us from a listener. Oh, he doesn't want anybody to know.

Steve: Yup.

Leo: I just hid it. But he lives in somewhere called Warner Robins, Georgia. And dang it, he's not changing his LastPass password: Hopefully I got your attention, but I wanted to share something that I think may have gotten away from us; and, if I am wrong, I would like for you, Steve Gibson, to correct me.

Steve: All right.

Leo: From everything I've heard, I don't see a reasonable reason for me to change my password. I followed the advice from LastPass. I have not used my password on any other site. I am using two-factor authentication. I did change the number of hash iterations to a random number. Thank you so much for that tip, and I have passed it along to everyone I know. So my point is, taking everything into account, it's a single password for the site, never used anywhere else. A strong password, greater than 15 characters, lots of entropy. Two-factor authentication. Oh, I use even - I use more than that.

Steve: I do, too.

Leo: I use, like, 20, 25. Why not? You know.

Steve: Yeah.

Leo: Two-factor authentication, random hash iteration, and the release of information from LastPass that someone may, MAY have gotten this information. For what feasible reason do I have to change my password? I think we security nerds constantly go for the most secure, when at times we only need to be reasonably secure. I know that can change in the future, but for now I'm not changing it. Besides, I think I have a lot more concerns with the fact that between Anthem and OPM, the Office of Personnel Management, yes, that most recent breach, 18 million strong and growing, letting my personal information into the wild, I've got a lot more to worry about with that.

Steve: And I completely agree.

Leo: Yeah.

Steve: The key was the second one. He said, "Strong password, greater than 15 characters, and plenty of entropy." Given everything we know, that LastPass's stored archive took that after your browser had hashed it like crazy, like 5,000 times there, they took it and hashed it with - and it's more than just hashing. The PBKDF2 is a fancy algorithm that is doing sequential operations and successively XORing the output into an evolving XOR. I mean, there's a lot to it, more than just hashing. And they do that 100,000 times. And he has a password greater than 15 characters with plenty of entropy. You have nothing to worry about. Nothing.

Leo: Yeah. And that's what I told my mom.

Steve: Nobody's getting it.

Leo: That's what I told my mom, yeah.

Steve: Yup, nobody's getting it.

Leo: Yeah. Security Now! listener Enrique suggests one mystery remains: Steve, you talked about the fob vulnerability - and your baguette - for the first time on Episode 407. Then in Episode 508 you provided a great explanation of how it's exploited. However, one mystery remains: Why do the crooks strike on the passenger side? Oh, I didn't know that. It seems like the hack should work on any door, but they seem to tap and target the passenger side.

Steve: Yeah.

Leo: Insert the usual here: Proud owner of SpinRite, can hardly wait to download the show to listen to it while I work around the house, blah, blah blah.

Steve: So there actually is an answer to the mystery. It is the case that they go in on the passenger side, because that's where the glove box is.

Leo: Oh, of course.

Steve: Because what they're not looking to do is drive away because they don't know what to do with the car. I mean, they could probably get one, but then what?

Leo: Yeah. They don't really want your car. They want your stuff.

Steve: Yeah. They want your stuff. It's creepy if they want your license and registration because then they know more about you than you'd like bad guys to know, if you keep that in the glove box. But that's, basically, they're going in the passenger side just to get what you may have in there because it costs them nothing to do it, if they've got the technology. And maybe they'll get lucky and get some goodies.

Leo: There you go. ChickenHead21 in the chatroom says it could also be that the transmitter is under the passenger side, and the steering wheel would be in the way. So it's just more likely that they can get in on the passenger side.

Steve: Given the fact that we're able to approach our car from any direction...

Leo: I think it doesn't matter.

Steve: That demonstrates that it doesn't matter. I did forget, though, our friend, our antenna expert, Spencer Webb, did a blog posting 29 days ago, when we were talking about Passive Keyless Entry Systems, and has a short YouTube piece. I just encountered it before doing the podcast, so I don't have links to it. But he has a very compelling demonstration, and fascinating, where he takes a high-quality silvered antistatic bag and demonstrates that folding it over, that is, folding the opening over a couple times is crucial to it functioning. He demonstrates. He has a PKES car, and he goes back and forth, like showing it different - oh, there, you found it. Let's play it.

Leo: Oh, all right. I'll turn it up here.

[Clip] SPENCER WEBB: I've been with Steve Gibson on a podcast, TWiT 255, from a couple years ago. And I have one of those cars that uses the keyless entry. So I wanted to do a quick test and show an easy solution that does work, with a caveat. So we're right up against the car. And if I have my fob with me, and it's in my pocket, I could just go over and unlock the car. Okay? So now let's start over again. I'm going to lock it. Okay? And if I very carefully pull on the handle without putting my hand on the capacitive sensor, it does not open. But if I put my hand on the capacitive sensor...

Leo: Oh, look at that.

[Clip] SPENCER WEBB: ...it immediately opens, and it unlocks.

Leo: I didn't know there was such a thing.

[Clip] SPENCER WEBB: Okay. So now we're going to try a shield solution. And this is an antistatic bag. I got these from McMaster-Carr. Let me show you the part number.

Leo: One of those mylar bags you see.

Steve: Yup. It's got a zip lock on the top.

[Clip] SPENCER WEBB: This is a McMaster-Carr part. It's a bag of envelopes. I think there's, I don't know, about eight or so of them.

Leo: Spencer's awesome.

[Clip] SPENCER WEBB: And they're for about 10 bucks.

Steve: Yup.

[Clip] SPENCER WEBB: And I'm going to drop my key fob into the bag. I'm going to seal the top of the bag. Okay? But I'm also going to roll the bag over, one fold, two folds. Okay? And now I'm going to do the same thing. I'm going to try to open my door.

Leo: Oh, it doesn't work.

Steve: Yup.

[Clip] SPENCER WEBB: It does not work.

Leo: Successfully blocked it.

[Clip] SPENCER WEBB: So this is an effective shield against the radio frequency that the car is using to ping the fob, to have the fob respond.

Leo: That's similar to the bag that FasTrak gives you...

Steve: Right.

Leo: ...to put your FasTrak in when you don't want it to work.

[Clip] SPENCER WEBB: Now, one important caveat.

Steve: Watch this.

[Clip] SPENCER WEBB: If we do not fold the bag over...

Leo: Oh, you've got leakage.

[Clip] SPENCER WEBB: ...and we just use the zip lock, okay...

Leo: Ah, there's leakage.

[Clip] SPENCER WEBB: ...on top, watch what happens.

Leo: Wow. Opened right up.

[Clip] SPENCER WEBB: Okay, it does work. So while we're trying to make a shield, the zipper closure does not allow this to become an effective RF shield at the mouth at the case. Even though it's mechanically closed, it is not sealed to the RF. But once we - I'm going to relock it. And now I'll ignore the zipper closure completely. I'm just going to roll the top over, one fold, two folds.

Leo: Look at that.

[Clip] SPENCER WEBB: Okay? And now it's effective. A Faraday shield is only as effective as its smallest opening.

Leo: Integrity, yeah.

[Clip] SPENCER WEBB: And that's why that fold is actually pretty important. A piece of aluminum foil would work just fine, except that it would be hard to reuse. This is a pretty simple solution. I'm Spencer Webb with AntennaSys.com. Thanks.

Leo: Thank you, Spencer. That was great.

Steve: Wasn't that cool? Yeah.

Leo: Yeah. So, yeah, those mylar bags would work, then. You just have to make sure you seal the top properly.

Steve: Yeah, exactly. And just like roll the thing up in it and seal the top, and then you're good to go. So thank you, Spencer.

Leo: Thank you. Oliver Widder, Hamburg, Germany cooked up an interesting question about HSTS, Strict Transport Security, which, by the way, we use on the website. Although it's an issue with subdomains. But I can explain that if you're curious. What would happen if I set the HSTS duration for my domain to almost forever? Ours was set for 400-some years, by the way - and then sold - we changed that - and then sold the domain to someone else? Would he be forced to abide by strict transport protocol for that duration, even if he didn't want to? Thanks for everything. Oliver.

Steve: Isn't that interesting. I had never really...

Leo: Yeah, I never thought of that.

Steve: Never thought of it, if you wanted to transfer the domain. And the answer is yes.

Leo: Wow.

Steve: There's no - well, okay. So if you anticipated the sale, you could change your HSTS...

Leo: That's what we did. We lowered it.

Steve: Right. Now, there might be some browsers out there, Leo, that have not come back since they picked up the 400-year value, so they're still 400 years. But you lower the value, and so that will update that for that browser's next visit, and bring it down to something reasonable. I looked around to see whether there was a maximum, and I couldn't find anywhere that said, oh, it's only good up to this length of time, because I was going to say to Oliver, if he set it to a year, then it's true that, if he sold it to somebody, that other person would have to honor that for a year. But once it expired, then they could back off. And of course, now that we've got Let's Encrypt coming, being secure is just not going to be a problem. And it is the case that you can edit that down. But only as browsers come will they get the newer, reduced duration.

Leo: So some people may still have our 400-year certificate.

Steve: Yes.

Leo: Yeah. So we'd have to honor that.

Steve: But that's fine, as long as, you know, they're going to come back sometime between now and, what, 2415.

Leo: I hope so.

Steve: Yeah. And in that case...

Leo: And if they don't, well, then, the heck with it.

Steve: That's right. You probably, you know, you've lost them as a listener, yeah.

Leo: Wow, that's interesting.

Steve: Or besides, their iPod will have filled up by podcasts by then. They won't have room.

Leo: Yeah, right. Isn't that interesting.

Steve: Yeah.

Leo: Yeah. So we had to - we initially put it in, but we have a lot of subdomains, including, like, our security cameras are on TWiT.tv.

Steve: Ah.

Leo: And we didn't really want them to have to use Strict Transport Security.

Steve: Nope.

Leo: So there's a directive.

Steve: So this is an argument, include subdomains, which is optional.

Leo: We took that out.

Steve: And you take it out, yeah.

Leo: Right, exactly.

Steve: I have the same thing.

Leo: Yeah. Dave McLain, Steelville, Missouri poses an interesting IPv6 question: Steve, I have a question about IPv6, NAT, and small network security. I was just reading your page about using a small router to provide NAT for a small network, how that provides a lot of additional security, even if there's only one computer attached. What I'm wondering is how is this going to work with IPv6? And once that's in common use, will an inexpensive router still do NAT? Or since there are so many more IPv6 addresses, is NAT is no longer needed? Maybe we won't do it anymore. Or will each machine get a unique IP address that's also visible on the open Internet? How, why, what is - what's going to happen, Steve? Dave.

Steve: This was something I had never thought to wonder before. But he's exactly right. I mean, as we know, IPv6 completely obviates the need for NAT. There are, I mean, it's so ridiculously many IPs that every single person can have, like, an entire Internet worth of IPs for themselves.

Leo: Right.

Steve: I mean, it's just crazy. When I talked to the providers of my T1s, because I wanted to get myself up to IPv6 - I have to have it here so I could develop a future ShieldsUP!, which is IPv6, which people are going to want someday. And they said, yeah, we'll give you, you know, a Class A. Would 64,000 IPs be good enough for you? It's like, what? For me?

Leo: Wow.

Steve: Yeah, okay.

Leo: For little old me?

Steve: Gee, thanks. So, I mean, so it is, it's a really interesting question. Will ISPs allocate blocks of IPs for customers? I don't think so. I think ISPs, even though they could give customers blocks, they just - they're ISPs. I mean, they're barely giving us bandwidth. So I think that we'll probably still have NAT. I think that there will - certainly IPv6 is really, from a NAT standpoint, nothing but instead of having a four-byte IP address, we have a 16-byte IP address. So the tables need to get bigger. But otherwise nothing else needs to change. So we get the security of unsolicited packets still being dropped at our WAN/LAN border. And inside we could be IPv4, if we wanted to.

If you had, like, devices that had not gone to IPv6, that's actually why I imagine we'll still have NAT, is there are probably older consumer devices, and maybe the IOT stuff. People are buying all kinds of Internet Of Things stuff, and they're all IPv4. When your ISP says, hey, guess what, we're going to IPv6, it'll be your NAT router that will be your own IPv6-to-IPv4 translator. IPv4 in your home, but you get a crazy 128-bit address from your ISP on the outside.

So I think that's probably the way it's going to go. I think nothing will change. But we'll have another generation of routers that will - and I imagine there'll be some overlap. They'll start coming out that are able to do both 4 and 6, maybe on either side. And then when the transfer happens, as things begin to ultimately move to 6, it'll sort of be painless.

Leo: Wow. We don't know when that's going to happen. Might be quite a while.

Steve: No, it's actually not going to happen, Leo. That's just the whole thing is...

Leo: That's what I think.

Steve: Nobody wants it to happen.

Leo: And so carriers, I mean, the ISPs are doing carrier NAT and eliminating the need for it, basically.

Steve: Yeah. I mean, exactly. We know that there are now ISPs that deliver 192.168 or 10-dot network IPs to their customers. I mean, the box doesn't care. It says, oh, okay, fine.

Leo: Yeah, looks good to me. I'll take it. Very nice. Dante Bertana in San Mateo, California, Steve's home town, that's where Mom was.

Steve: Yup, San Mateo.

Leo: Is Mom still there?

Steve: Mom's there, and my sister and her family are there, yeah.

Leo: Oh, that's nice.

Steve: So that's where I go visit.

Leo: He is worried about buffer overflows. Who isn't? My question is, he says, how exactly do buffer overflows lead to exploits? Oh, that's good. I don't understand how simply crashing a program or app leads to an exploit vector. I think you've explained this in the past on the show, but maybe you can give me a 30-second refresher? Thanks, as always. I've been listening since Episode 1. I was 13. And I look forward to every new episode. What does that make him now, 23?

Steve: Yeah. Cool. Okay, so Dante...

Leo: We've said this many times. I've said it, too.

Steve: Here's the short answer. I was thinking about how could I say this in a way I never have before? And that is, due to the architecture of the CPU, the actual registers and memory arrangements, there's something called "the stack," which is a - it's sort of like an accordion. You're able to borrow memory from this stack in a sequential fashion and give it back sort of in the reverse order. And it's very fast, that is, the operating system doesn't need to be involved. So you don't have to make OS calls. So it's fast, and it's there, and it's convenient.

The problem is we use the stack both for program addresses and data. So you can, like when you're going to go somewhere, and you want to be able to come back, you put the address of where you are on the stack. Then you go somewhere. And in order for that subroutine, for example, to come back to you, it knows to get the address from the stack. The problem is, while it's doing something, it has access to the stack, sort of everybody does, in a single process. So, or actually on a thread of a process, which now it's getting a little more complicated.

But essentially what this means is, in one place, we are mixing in addresses that we're going to jump to for code and data because one of the things that happens is the so-called buffer overrun. We create a buffer dynamically by just saying I want some buffer space on the stack. And we do it because it's so fast. When we do it right, there's no problems. It's when we do it wrong that there's a problem. And if the program asks for a certain amount of buffer space, it's sharing that space. Maybe like on both sides are pointers to other code that we're going to go back to. So if that buffer is overrun, if we store more on the stack than we thought we wanted stack space for, then we overwrite the instructions, like right after the buffer, that are telling us how to get back to where we came from, the return address.

Now, if we overwrite it randomly, then the processor sees an address, and it doesn't know if it's random, I mean, it doesn't know there's anything wrong with it. It follows it blindly. And boom, crash. And that's where we start turning this into an exploit. Because once a hacker realizes that they have a way to put too much data in, then they start to look at it closer, and they say, hey, you know, it's this particular data that the processor treated as a jump destination. So if I have some other code somewhere else, or if there's some in the operating system that's not supposed to be executed, but if I can put data there that points to it, then it will get executed. And that's how we escalate from a crash to an exploit that actually does something that the hacker wants.

It's a side effect of the underlying architecture that's, like, fundamental to our computers. It didn't have to be that way. If they'd been designed with security as an absolute paramount aspect, they would have never mixed data and instructions. That's the danger, is we're mixing data that the user can control and that comes from the outside. And within the same place we have jump instructions, or addresses that are taken as destinations of jumps, right there next to the data. And it's just dangerous.

Leo: And this is where we went wrong with imperative programming, side effects, memory allocation. If we'd just stayed with LISP, none of this would have been a problem.

Steve: You know, it's those darn engineers that designed the CPU. They thought, hey, yeah, look how convenient this'll be. It's like, ooh, yeah, but, boy, if we make one little mistake, it overwrites the stack, and then all hell breaks loose.

Leo: Yeah. But you see why people are doing garbage collection and stuff. You know, malloc is not your friend.

Steve: No.

Leo: Unless you're used to it. But, see, then you're - that's the thing. You're suckered into it. Ray Haynes in Oceanside, California - actually, I wonder, I guess LISP is stack driven, so you can still screw up the stack.

Steve: Oh, boy. LISP invented the stack.

Leo: Yeah, yeah. Roy Haynes in Oceanside, California wonders whether LastPass is really TNO: Steve, I know you and Leo are fans of LastPass and have fully vetted it. Well, Steve has. I'm just following along for the ride. But I believe in Security Now! 512 you used TNO to describe LastPass, which has shown not to be true with network breach. If it were indeed TNO, you wouldn't care what data was taken. In fact, you are trusting them with secrets. I use 1Password with a long password, and for sharing I put the files on Dropbox. I don't trust Dropbox to keep my files safe, but I do trust the encryption of those files. Is this not true TNO versus LastPass's dependence on network security? Keep up the great work, and you, too, Leo. Ray Haynes, Oceanside, California. I'm glad you put this in because I'm hearing this from a lot of people.

Steve: Yeah. It's sort of a good question. I have to agree...

Leo: What do you mean by TNO, first of all?

Steve: I have to agree that the need to synchronize devices in the cloud requires us to soften our security to a level that's super strong but, I'd have to agree, not absolutely TNO. It's like how, if you're going to use Dropbox in a way where you could get your data from some other web browser, well, that's not TNO because you've had to enable them with some information where they could decrypt the data on your behalf. And that's really what we've done with LastPass. We've said, we want you to store this blob in the cloud, but we're going to need you to decrypt it on our behalf in order to synchronize our devices.

So I have to agree with Ray that this is, I would argue, there isn't - there is not a way to make this stronger. That is, what happened was someone may have gotten into some data that they were trying to protect and, I mean, did as good a job protecting as we could expect anyone to. But the fact that it did create an opportunity to decrypt a password says, and I agree with Ray, that if it were TNO, the difficulty would be 2^256 because that's what we do. We give data blobs to be held which we have encrypted with a key, a high-entropy key, completely random, with an entropy of 2^256. That's the only thing they have.

LastPass, in order to do what we're asking them to, has a much weaker key. That's why we've talked about, if it were a really high, super high-entropy password, then you don't need to worry because it's the lack of entropy in the password which, due to either it being too short, in which case it just can't contain, doesn't have enough characters to contain a lot of entropy, or if the character set is weak, or the characters are predictable, all those things lower the entropy.

But we are giving, because we take a password, we hash it, then we give it to them, they hash it, that's still less than 2^256 strength, in which case I would agree that it falls - TNO is 2^256. Meaning that there isn't a shortcut that would allow anyone to do anything, a hacker, on average, half of that many brute-force guesses in order to get the result. And this is certainly - it's a lot, but it's not 2^256. So I, yeah, I agree. We have had to back off from TNO. So I used the term incorrectly last episode. I agree.

Leo: TNO, and this is true of Dropbox, it's true of LastPass, eliminates some uses that people would like.

Steve: Right. There is a tradeoff. If you're absolutely not trusting someone, then you've completely removed a set of functions, many which you may want for convenience. Perfectly said, Leo. And in this case, we want LastPass to be able to synchronize our devices. So we have to give it the ability to do that.

Leo: And we should point out that, if he's using 1Password, and 1Password allows him, which it does, to use his password store on multiple devices, it's got the same problem.

Steve: Right.

Leo: By the very nature of it. Anything that would allow you to open your secure store on other devices would be ultimately not TNO because you need that password that unencrypts everything; right?

Steve: Right.

Leo: And that password, once it's shared to a third party...

Steve: Yeah, I'd have to think this through because, if the blob were in the cloud...

Leo: I thought LastPass couldn't get into our blob. But of course they're - they can't, can they, because they don't...

Steve: They can't. They have to crack it the same way the guys that may have stolen that information would have to crack it. I have to think what it is...

Leo: So that's TNO, isn't it?

Steve: Well, no. I have to think what it is. There must be some things we're asking them to do that require this. Because if all they were doing was synchronizing a blob, then all of our different devices could keep the key, and the blob could be fully TNO. So it must be that there's some features that we've come to expect that have required that softening.

Leo: I'm sorry. You need to explain to me why LastPass isn't TNO. I thought TNO means that the data I'm storing on their servers, they can't get into.

Steve: Correct. And they can't, any easier than the hackers may have been able to.

Leo: Right, right. So that's TNO. Why is it not TNO? Why is it not Trust No One? I'm not giving them keys to anything that lets them look in my data.

Steve: No. But they did lose something. And in TNO, there's nothing that they can lose that could compromise you. So, for example, if they weren't doing that extra 100,000 hashing, then they would be holding for us, apparently, the result of our browser's hash.

Leo: Right.

Steve: But why do they have to hold anything? I don't understand. It must be for recovery or email or something. I mean, the point is they don't actually have to hold anything. And then they would be TNO. But they are holding something and protecting it strongly. But that's no longer TNO.

Leo: Is it so that we can log into our account? They can't - we should be clear that the hashing they do is one-way. They cannot ever get our password.

Steve: Right. Okay.

Leo: Our master password in the clear.

Steve: Okay. So imagine that they didn't have this, that they had their database of people's encrypted data. And imagine that that...

Leo: We'd have to validate so that they would know to give me that database. Right? So they need my master password in some form.

Steve: No.

Leo: No.

Steve: So imagine that they have a user's set of data.

Leo: Yeah.

Steve: And the bad guys know that the user browser hashes a password 5,000 times. Then the bad guys can do a brute-force on the key, if LastPass let that blob loose.

Leo: Right.

Steve: That is, the problem is that, well, but there may be another - I have to think this through. I mean, this stuff is really complicated. Because it might be that the key is not directly derived from the user's password after hashing. That could be used to encrypt a high-entropy value. I just - I've forgotten some of the, I mean, this really depends upon the minutiae of the architecture.

Leo: So it may not be technically Trust No One. But it's effectively Trust No One because of the 100,000 iterations on the hash if you used a strong password; right? To me, Trust No One, look, this is what it means to me. Maybe I'm - it means that I'm not - that no one at LastPass can look at my stuff. They don't have the information they need to decrypt my stuff, any more than the bad guys do.

Steve: Okay. Here's the problem. Here's the way to say it. Any system that we have - and actually this was the thought I had when I was preparing this. I don't know why I didn't come up with it quicker. Any system that we have starts with us authenticating and then going through some path that allows us access to the data. And if that's not a secret, and we know that secrets cannot be kept, that is, we don't depend upon secrets. So somewhere there will be a password or, I mean, that's what we're using for LastPass. There's a password. We have it. The browser hashes it. We give it to LastPass. They hash it. But my point is, there's a pipeline where at one end of the pipeline is a password.

Leo: Right.

Steve: That is the weakness.

Leo: Right.

Steve: Because that means that, if the pipeline is known, and as I said, it is known, it is not - we know that, for example, cryptography does not depend upon knowing what your algorithm is. It depends upon a key of a known algorithm. So if the pipeline from the user's password all the way to the encrypted blob is known, anyone can brute-force that password, through that pipeline, no matter how you design it, and end up with your results.

So what you want to do is what LastPass has done, and that is, make this pipeline as painful to move through as possible. They don't want information, which is why we encrypt it. We first hash it like crazy in the browser. And then they, because they really want to protect us, hash it insanely, another 100,000 times, to make this pipeline incredibly painful to brute force. But ultimately, it comes down to, if you understand the pipeline, you can put guesses in and test them.

Leo: Yeah. It's simply not TNO because we give them something that was created out of the password. We give them the password. And that makes it not TNO. The only way it would be TNO is if we encrypted locally with a password that only we had, and then that password wasn't shared. Right? That would keep it TNO.

Steve: No. No. Well...

Leo: No?

Steve: Okay, it would keep it TNO from them. That is, okay. But a bad guy, a bad guy could still guess our password.

Leo: Right.

Steve: And give it to them.

Leo: Right. So anytime you put anything in the cloud, it's now inherently not Trust No One. Right? I mean, I think we've now made Trust No One very hard to live up to.

Steve: We really have. We've raised the bar.

Leo: I mean, anything in the cloud, you're saying, if a bad guy could get those and work on them, then they're not trust...

Steve: And then reproduce what the user does for their own access, then...

Leo: Right, then it's not trustworthy.

Steve: Then they've got the data, yeah.

Leo: Even if we hold the password. So your previous form of TNO, which was pre-Internet encryption, or PIE, even that's not TNO in this definition because he can get the data and then reverse our process by brute force.

Steve: True. True.

Leo: Well, now you've made the definition useless. In other words, it's like Ben Franklin.

Steve: Well, it's Episode #513, so it's, you know...

Leo: A secret could be kept only by two people if one of them is dead.

Steve: That's right.

Leo: Trust No One only means you trust no one. And to store anything or any, you know, that's not going to be very useful to anybody. You didn't - we haven't really answered why. And I'm sure that LastPass could, why they want you to put a password on their service. But I think that that's so they can validate you.

Steve: Oh, that's the only means we have of...

Leo: And give you the store. You have to authenticate.

Steve: Yeah. It's the way we have of authenticating ourselves to their service.

Leo: Right. And the point being made is that they're not taking their copy of your unencrypted password and comparing it to your unencrypted password. They're taking their 100,000 times hash of their password and comparing it to the 100,000 times hash they made of the one you entered and seeing if that matches.

Steve: Right. And that's why they're doing so much work on our behalf.

Leo: Right, right.

Steve: Every time we have a...

Leo: So they don't know your password.

Steve: Every time - nope.

Leo: And they don't, can't unencrypt your data any better than anybody else can except through the same bad guy...

Steve: Exactly.

Leo: So I consider that TNO. I'm sorry, you've made it very tough to live up to.

Steve: Yeah, it is.

Leo: But if you're going to store stuff in the cloud, this is as good as you can get.

Steve: Yes. And I think the easiest way to see it is that, if a user has access to their data, using a password...

Leo: Right.

Steve: See, and this is where TrueCrypt is tricky because TrueCrypt, remember, can tie the keys to data files and other...

Leo: Right.

Steve: You're able to mix other things in. There, there's no brute-force attack. But here we're just using something we know. And so if we just need something we know, and that gives us access, bad guys can guess what we know and eventually get it.

Leo: Okay. In millions or trillions of years.

Steve: Yeah, exactly. In, like, never.

Leo: And in some impracticable fashion.

Steve: Yeah. Before this other guy's HSTS expires in the year 2415.

Leo: Steve Gibson, this is why we love this show. It's these brain twisters. Who needs Loop? We've got you, Steve. Steve Gibson is at GRC.com. That's where you go to find his great utilities, including SpinRite, the world's best hard drive maintenance and recovery utility. All the rest are free. There's so much great research and information up there - SQRL, Perfect Paper Passwords, Password Haystacks, I can go on - Vitamin D. I can go on and on. But the best way to do it is just go to GRC.com.

If you have a question for Steve, that's one place you can leave it, GRC.com/feedback. Don't email him. You can't. His address is Trusts No One. But you can also tweet him, @SGgrc. And he does read those. He also puts 16Kb versions of the show, 64Kb audio. He puts up transcripts, good human-written transcripts, after a few days, at the website GRC.com. We have also copies at our website, TWiT.tv/sn. You can subscribe. Show notes are at GRC.com. We have a link to that at TWiT. And on and on and on.

We do this show Tuesday afternoon, 1:30 p.m. Pacific, 4:30 p.m. Eastern, 20:30 UTC, live.twit.tv. We're going to keep that up and running for a while because I guess there are some people who can't use JW Player. Their work blocks it. So for those of you in the military, we're going to keep live.twit.tv running for a while till we figure out what to do. But you can also go to the new page, which is prettier, TWiT.tv/live. And we will see you in three weeks.

Steve: We're going to see you in three weeks.

Leo: Yeah.

Steve: Everybody will be seeing me every week.

Leo: Yeah. Who's hosting with you? I think Father Robert?

Steve: I don't know. I would - I think probably the Padre, yeah.

Leo: Yeah, yeah. So we'll still be on for the next three weeks. But I will be in Europe, paying no attention.

Steve: Dangling your feet overboard.

Leo: I'll be dipping my tootsies in the Rhein. All right, Steve. Thanks for being here. We'll see you next time.

Steve: Thanks, buddy. Right-o.