The Windows AtomBomb

Description: Leo and I discuss the answer to last week's security and privacy puzzler, Let's Encrypt Squarespace, the new open source "LessPass" app, LastPass goes mobile-free, many problems with OAuth, popular Internet services' privacy concerns, news from the IP spoofing front, Microsoft clarifies Win10 update settings and winds down EMET, a hacker finds a serious flaw in Gmail, MySQL patches need to be installed now, a tweet from Paul Thurrott, a bit of errata, and the Windows AtomBomb attack.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-585.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-585-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Wow, there's a whole bunch of stuff to talk about: the Patch Tuesday update, which came out today from Microsoft for Windows 10. We can talk about LastPass, now free for some; LessPass, maybe not as good as it sounds. And Steve will explain this whole Windows AtomBomb exploit. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 585, recorded Tuesday, November 8th, 2016: The Windows AtomBomb.

It's time for Security Now!, the show where we cover your security and privacy online with the king. My god, this man is the Explainer in Chief, Mr. Steven T. Gibson. The "T" stands for Tiberius. Hey, Steve.

Steve Gibson: Ah, that would have been actually very fun if Dad had thought to do that. But he gave me - my middle name was his first name, which was common back then. But that's fine.

Leo: That's fine. Now, this is Election Day. Before I let you do the show I have to make sure that you've voted.

Steve: Oh, my god, yes. In fact, I'm annoyed that I forgot before we started recording to get the sticker off of my absentee ballot paperwork because I noticed it was down there in the lower right-hand corner.

Leo: Yeah. I didn't do it, either.

Steve: But I didn't peel it off. And it's like - because I want to proudly say that, yes, I'm a member.

Leo: You want to wear the sticker, yeah, yeah. It's okay. We voted a while ago. Our votes are already in the system.

Steve: As did I, yup.

Leo: Yes.

Steve: Yeah, I'm a permanent absentee, just because it's like, you know, you get the ballot a month before, you can study it.

Leo: Right, it's convenient.

Steve: I normally sit down with some friends, and we all go over it together and figure out what we want to do. It's sort of a nice little social time.

Leo: As I mentioned last week, we did a special episode with Ron Rivest on voter security. And one of the things Ron said, though, is early voting's great, but it should be poll-side voting. Mailing ballots is problematic. Sometimes ballots don't make it through. In fact, I heard somebody got his ballot returned because the machinery misread the address. You would think the postal service would handle those ballots with kid gloves, but some mistakes do happen. So he said the best thing to do is do that, but then take it to a polling place. And here in our area there are places I could have gone last week and the week before to hand it in personally.

Steve: Right.

Leo: And that's the recommended, for security, that's the recommended way to do things.

Steve: That certainly does make sense.

Leo: Yeah, yeah. Anyway, what are we going to do today?

Steve: Well, so as promised last week, which began with "Next Week on Security Now!," we're going to finish this podcast with a discussion of a very clever use of existing Windows APIs that allows an attacker to execute their code in a victim process. So that's

not something you want to allow. And normally the way that's done is with the famous buffer overrun.

It turns out a very clever security engineer researcher figured out how to do the equivalent of a buffer overrun without breaking any rules, not relying on any misbehavior of any kind. And he called that the AtomBomb because it uses something known as the "global atom table," which is a feature we'll discuss that Microsoft built into the Windows API back in Windows 1.0. It's always been there. And the point is it turns out it's fundamentally insecure, and they can't remove it because lots of applications that exist depend upon it. So very, very intriguing. And, now, there are some mitigations we'll talk about. But a great topic for the podcast.

But that's on the heels of another crazy week. We've got the answer to last week's security and privacy puzzler. Squarespace decides to encrypt. A new open source app, unfortunately named LessPass - I hope he doesn't try to get a trademark on that because LastPass's attorneys are going to stomp on it.

Leo: I kind of like the solution, though. I'm curious what you think about it because it is open source, yeah.

Steve: Yes. We have LastPass going mobile free, which confused a lot of people. They all think that means it's free. So we'll clarify that. A surprising bunch of problems with OAuth, both concept and implementation, which we need to cover. Some new Internet services' privacy concerns, and some of the details are sort of gut-wrenching. News from the IP spoofing front.

Microsoft is clarifying the settings for Windows 10 Update and declaring they're winding down EMET, their enhanced mitigation technology, saying that it's really kind of obsolete now. A hacker finds a serious flaw in Gmail. The need to make sure, if you're using MySQL, that you have patched it because some patches have gotten out operating proof-of-concepts, and they're only about a month old, so you want to make sure you're current. A tweet from Paul Thurrott that I got a kick out of. A little bit of errata, and we'll talk about the Windows AtomBomb. So buckle up, baby.

Leo: Wow. Quite the table of contents.

Steve: We've got a great Picture of the Week. This shows a salesman standing in front of a couple. In the background is a big banner sign that says, "Connected Appliances." We've got washers and dryers and little microwaves on the side, and they're standing in front of a bunch of refrigerators. And the caption on this cartoon says - it's the salesman speaking, saying, "Can I interest you in a firewall for your toaster?" Yes, the future. The future, folks.

Leo: You know, I had somebody call the radio show. He had bought a Samsung refrigerator five years ago. And, you know, we think of the timeframe of a refrigerator, 10 years? You keep it a long time; right? But the problem is it had a big - not actually at the time, it wasn't that big, but it had a screen, and it had Calendar and stuff on it. And it had interfaced with his Google Calendar. But Google changed the API, and Samsung has never updated the software.

Steve: Wow.

Leo: So he called and said, you know, "How can I get my calendar back on my refrigerator?" I said, "You can't, you can't."

Steve: Wow, that's a perfect example of the state we're in at the moment. I haven't said this yet, but I'm beginning to think that the only way this IoT thing is going to settle out is if we end up using the existing major proven security-conscious hubs, like Apple and Google. I think, you know, what they need to do is create an open API. And Apple may be a little challenged to do that. We'll see how that evolves. But from my standpoint, right now there are no standards. As we've talked in the past, there are some standards sort of trying to evolve. But I think the way to do this is to allow consumers to purchase all of the gizmos and gadgets they want, but have them communicating through a responsible party, someone like an Apple or a Google.

Leo: Good, good, yup.

Steve: Who can invest in getting it right. The problem is people are getting cameras and light bulbs and door locks, and there's complete fragmentation. The traffic egressing from their home now carries separate streams for all of this stuff, you know, blasting across the Internet with lord knows what and literally unknowable sets of vulnerabilities.

Leo: I don't know if this is new; but the story came out, now maybe there was something from DefCon about a flaw in ZigBee allowing the Hue light bulbs to be hacked from 300 feet, 350 feet away, and then the worm spreading?

Steve: Yeah, actually we talked about it months, yeah, we talked about it months ago.

Leo: Yeah, that's what I thought because it sounded like that story. So this is just somebody who's found it and repeated.

Steve: And that's a perfect example. As long as, well, you know, I think the argument is this won't get fixed by itself because consumers can't be bothered to care, and manufacturers don't need to care in order to sell their stuff. You know, people just don't understand the threat that they're under when they're putting cameras in their home and thinking how wonderful it is that they can see what's going on in their home when they're away. Well, we've covered specific instances where, yeah, and so can everybody else see what's going on in your home when you're away. Which is not what you had in mind.

Leo: Yeah.

Steve: So anyway, last week's security and privacy puzzler asked - it posed the question that actually came from a listener that I thought was great. We know that common security practice dictates that compressible data should be compressed before it's encrypted because after any data has been encrypted, it by definition will not be

compressible because encryption essentially removes the entropy. It removes the patterns. It removes any of the hooks that compression uses in order to squeeze out space.

So, for example, a perfect example is if you had in a file a 4K bit of zeroes. Any compressor worth its salt will look at that and go, whoa, wait a minute. And instead of sending out 4,000 zeroes, it'll put out one zero and a "repeat this 4,000 times" clause. And so, as a consequence, two little tokens go out across the wire; and at the other end, the decompressor understands that, interprets that little token, and then re-expands it to its original size.

Well, the problem is that 4,000 block of zeroes, if it's encrypted, it becomes noise. Any good encryption will turn something like that into something that's indistinguishable from any other data. So that can't be compressed. So the question we posed was, does compressing, then encrypting, which we know is what you have to do if you're going to get any compression, create any threat to security or privacy? And I guess it was sort of a rhetorical question because we know the answer is yes.

Now, we have covered in the past two famous and classic attacks on compressed HTTPS sessions. And that was the CRIME and the BEAST attacks, where the researchers realized that, because the headers of queries were being compressed and encrypted, if they had a means to insert some of their own data, to add some data into the headers and then have it compressed, and if they could see the result after compression - so they had to have kind of a strange position on the network. They had to be able to monitor, passively monitor the communications as it was leaving the system, but also have a means of affecting what was being compressed.

So it was a theoretical attack. It took a huge amount of time, a crazy number of queries. But it wasn't infeasible. But more than anything it demonstrated, okay we have a problem here because this shouldn't be allowed to happen. And all it took was the browsers to essentially introduce their own little kludge to work around this potential problem. And that kludge served to obscure the information that this attack was keying on. So we've seen it before.

The reason this came back up - and I found just a beautiful example of this - is some researchers recently, at the University of North Carolina at Chapel Hill, have extended this concept by successfully attacking encrypted VoIP, that is, Voice over IP communications such as Skype, Fring, Google Talk, and any SIP-style encrypted comms. It turns out that, in order to get real-time interactivity, in order to have minimal delay, which is what you want in an interactive conversation, the VoIP, the digitized voice, is encrypted in packets, very short packets, so that they're able to then immediately be stuck on the wire and get to the other side to be decompressed, or I should say decrypted, decompressed, and then heard.

Well, the show notes have just a beautiful, frightening diagram from their research paper, where they demonstrate - and they pulled this off. Taking nothing but the packet lengths, they are able, through a series of steps, to determine what was said. And the reason is that it turns out that different vowels and consonants and sibilance and so forth have characteristic sizes, and they compress at sort of well-defined known rates, that is, percentages. And it's possible just from the size of the packets to reverse-engineer what was said.

So, now, once again, if somebody actually wanted encrypted VoIP, there are several things you could do. You could sacrifice the responsiveness, a little bit of the real-time speed, by grouping in blocks rather than sending individual variable-length packets out

for the sake of speed. The other thing you could do would be to deliberately pad the packets with a pseudorandom, where the pseudorandom key is known only to the sender, you know, just generated on the fly. You just grab a random number. That drives a pseudorandom number generator that decides how much fluff to add to the packet. The problem, of course, is that that tends to increase the overall bandwidth of the conversation. So there is a cost. But it would solve the problem of this being otherwise essentially decryptable or interpretable without decryption, just using the size.

So anyway, I love this as a classic example of why there are applications where you cannot safely compress, then encrypt, just because of the nature of the data you're compressing and the way it's available. Again, if you were to record an entire conversation in VoIP and digitize it, then compress it as a block and then encrypt it, no problem because you wouldn't be getting the little essential time markers of when these utterances were made and how long they each were.

So it's not that it's generally bad to do this. It's just that this was done without fully thinking it through. And in general, and we'll see more examples of this this week, it feels to me like we are sort of moving - and we talked about this last week, Leo - to sort of a next generation of the degree of effort being put into finding problems and demonstrating that things that we've been using in the past aren't as secure as we thought they were. And of course that's all for the good.

So today is November 8th, Tuesday. The 1st was last Tuesday. That makes this the earliest possible Patch Tuesday to occur in a month. It can't ever happen sooner than the 8th. On my Win7 system there was 134MB blob, the single security monthly quality rollup for Windows 7 for November. And so my system installed it. I did want to mention that last week we were wondering, I was wondering whether Microsoft would have had time to get the fix in which we discussed last week. This was that zero-day flaw that Google informed Microsoft of just, I mean, like without much notice. And after seven days, Microsoft's policy is we're going to let everybody know about this because this is being exploited in the wild. Microsoft did get that fixed, so that has been fixed as part of this.

So what users should know, our listeners I'm sure are always installing patches, although we have often talked about how in some cases it's inconvenient to do it at the moment. This is one you don't want to put off too long because it is fixing a known, currently being exploited, zero-day flaw that we discussed last week. So good news is it's been fixed.

Leo: Whew.

Steve: Yeah. So on October 24th, about two weeks ago yesterday, Squarespace had an announcement. And a bunch of our listeners, knowing that Squarespace is a TWiT sponsor, made me aware of it, and it just slipped through the cracks for the last two weeks. So I wanted to just note that Squarespace has adopted Let's Encrypt.

Leo: Yay.

Steve: Yes.

Leo: And it's even easier than really - because I just did it myself. You can see if you look at my LeoLaporte.com, HTTPS.

Steve: Yay. Cool.

Leo: But it's so - you don't have to do any Let's Encrypt stuff. They're doing it all behind the scenes. All you have to do is check a box.

Steve: Right.

Leo: It's awesome.

Steve: Yes. On their blog announcing this on the 24th they said: "Secure Sockets Layer, or SSL, is a technology that secures the connection between your browser and the website you're visiting. It allows you and your website visitors to feel confident that their information is secure. And we believe that confidence is an important part of your online identity.

"So, starting today, we're proud to offer free SSL on all Squarespace websites. Website owners should not have to pay extra or wrestle with complex technical issues to offer this basic security to their users. Every website can enable SSL, which will automatically direct users and search engines to a secure version of that site. The result is that millions of more domains on the Internet will be secured via SSL, our customers can take advantage of the confidence that secure websites bring users, and we will have helped the Internet take a huge step forward in promoting security by default.

"As an added benefit," they wrote, "websites hosted on Squarespace may enjoy a boost in search ratings." And we have heard some grumblings, I don't even know if it was put into practice yet, but we know that Google was thinking at some point they're going to promote encrypted sites over non-encrypted sites, just as another signal for their site ranking to pour into the pot as it's deciding what to do.

And finally they said: "Squarespace is taking care of almost everything, making this an easy transition for customers. To seamlessly manage SSL certificates for all of our websites, we've partnered with Let's Encrypt, a free and open certificate authority run for the public's benefit that provides free SSL certificates. Current Squarespace users can see instructions on how to enable SSL on their site by checking out our Help guide." So, yay. That's a nice step forward.

And I also encountered a couple sample screens of the upcoming, I think it's next year, but next year's around the corner. It's in early 2017, Google has said, and I saw sample screens from Chrome, Google's browser, where they're going to start criticizing non-HTTPS connections. That is, rather than saying nothing, they're going to say "Secured" when it is, "Not Secure" when it's not. And then there was a red one. I think "Not Secure" is gray. "Secured" is green. But there was a red one that was like, if there's some problem with the site of some sort that Google's not happy with. So, yeah. So clearly no site wants to be in a position where Chrome, now the majority browser on the Internet, is saying "Not Secure."

Leo: Yeah. That kind of made me mad. But you know what, in the long run, we all benefit.

Steve: Yeah. Everything Google does like this makes us mad for a while. And then we go, well, yeah, I guess it was necessary.

Leo: All right. And I should point out we didn't - it's a sponsor. Squarespace is a sponsor. They're not sponsoring right now, this show. And I just showed you real quickly, you go to your Settings > Security & SSL, and there's just a new setting in there. And your choice is "Insecure" or "Secure." You don't have to do - see, when I first read Let's Encrypt, I thought, well, how - am I going to have to run a script in my - I don't understand. But no, no, it's simple. Secure or insecure, boom.

Steve: Just turn the switch on.

Leo: And you get HTTPS.

Steve: It's interesting, too, because I'm careful not to say "insecure." I say "not secured."

Leo: That's a better way to put it.

Steve: Yeah, because that's the correct way to say it. But I understand what Squarespace - this was an investment on their part. They want everybody to turn that on.

Leo: They tweeted it a couple of days ago, and a lot of people said - because they said "SSL," and people said, "Oh, no, you meant TLS." Is that a pedantic distinction? Or is there actually...

Steve: Okay.

Leo: "I hope you meant TLS, not SSL."

Steve: They're right. You're right. So we have abandoned SSL. But the legacy term is still there. And I think they probably used it on purpose because it's more commonly known.

Leo: Right.

Steve: I would imagine that, I mean, I know that our listeners know what TLS is. And in

fact, even SSL 3.0, which was the last SSL, has been now formally deprecated in favor of at least TLS 1.0, and you should really have 1.1. So, I mean, we have moved to TLS. But for this purpose I could see why they would do that. I think I would have instead said HTTPS because that's the visible thing that typical Squarespace visitors and even users would be more familiar with. And it's the "S" of HTTPS that matters, rather than really the underlying protocol.

Leo: And they, I mean, it's Let's Encrypt, so we know it's the best; right? I mean, it's not - because it's Let's Encrypt, you can't use an insecure algorithm; right? I mean, they're using TLS.

Steve: Oh, I'm sure it's going to be, yes, yes.

Leo: Yeah, yeah. Yes.

Steve: And it's Let's Encrypt. And it's, what, you know, a year old. So it's absolutely going to be state-of-the-art.

Leo: Yes, right. So really all you - I think you're right, HTTPS. Although I notice in your notes here it says SSL.

Steve: Yeah.

Leo: So, you know, what are you going to do?

Steve: Yeah. And I did copy what they said, which was SSL. So, LessPass. First of all, I was confused when people started tweeting me the news of LessPass. And I'm thinking, okay, is that a snarky blog posting about something wrong with LastPass, which is what it sounded like? And the very fact of that confusion is a problem because that is the definition of a trademark violation, I mean, a trademark collision. So this is free and open source, but he'd better not try to get a trademark because he can't. That's the problem with something that is deliberately confusing and sounds alike. I mean, so for example, any attempt to trademark this would get violently stomped on by LastPass's intellectual property attorneys. So I would have advised this person, if he'd cared, not to call this thing LessPass. I mean, it is confusing. Okay, so...

Leo: He's not charging for it, I don't think; right?

Steve: No, and it's not commercial.

Leo: Yeah.

Steve: So, you know, but again, it's just, you know, it's inelegant to have an obvious

confusion with what is arguably one of the most popular, if not the most popular, well-known browser password managers. Okay. So this is something similar to what you were using a long time ago, Leo.

Leo: Yeah. Super Password Gen, yeah.

Steve: Right. And we've even considered it for SQRL because, okay, so let me step back a bit. What this does is it's a hash-based deterministic password generator. So you have three fields: the site's domain; your login name at the site, which is often your email address; and then your master password. Those three things are just poured into a hash. And I didn't dig into the technology much because it's not hard to do, and I'm sure this guy did it right.

So the point is that the result is dependent on every single character of the site's domain, your login name, and your master password. And it produces then probably a 256-bit, or maybe more, blob. The problem is then you need to format that, since you can't submit a binary blob to a website, as we know, they want passwords. So then there's a series of checkboxes. Do you want to have lowercase alpha, uppercase alpha, digits 123, special characters? What length do you want? And then he also thoughtfully added an integer increment modifier, which is a nice touch. And I'll explain what that's for in a second.

The result of all of that, when you've set all that up, it gives you a password. And the point is it is absolutely deterministic, that is, because it's based on - because what comes out the other end is based on all of those settings, if you later input exactly the same settings, you get exactly the same password in response because it's just based on a hash with then a bunch of output formatting to turn it from a binary blob into a password. And the reason it came up when we were looking at SQRL is that we know that one of the challenges that SQRL or any new authentication technology faces is that it requires support from the sites you want to use it on.

You know, the argument for SQRL is that SQRL's technology is very simple to implement on the server side, and there's already lots of packages that are in process. Many of them are working now, in fact. But the problem was all those other settings, that's what stopped us. And it's what would stop me from considering LessPass instead of LastPass. And that is, you would be advised to always - you have to always set the post-processing parameters the same, or you're going to get a different password. So you always have to say upper- and lowercase, and you want digits.

Now the question is, ooh, wait, did I, when I created the password for that site last month, did I use special characters or not? And I don't remember what the length was. And that's a problem because, as we well know, websites don't have standard policies for password acceptance. People are constantly tweeting me their discoveries of some otherwise hopefully secure site that says, oh, passwords are limited to 12 characters. So the problem is, if you set it at 20 because you're security conscious, now you encounter a site that says, no, sorry. Some sites don't tell you why. You know, we've talked about that, where there are instances where you just have to keep guessing until it finally accepts it.

So this is the problem with that is that I like the idea, I mean, very much, of a deterministic password. But because there isn't a universally accepted standard for what is an acceptable password, and the fact that, what would there be, the greatest common multiple, or least - I don't know what it would be, the least common factor, something, to

find the one that all sites would accept? Actually, there probably isn't one because some sites say you must have at least one special character and one digit and one uppercase and one lowercase; whereas some other site says, eh, no, you know, can't have any special characters. So this is a nice idea, but I just don't - I think it ends up being a lot more burdensome on the user than something like LastPass, where you can negotiate with a site to get a good password. And if it rejects you a few times, you just keep trying until you find one that works, and then LastPass remembers it for you. And you don't ever have to bother with this again.

Oh, and one last thing, the integer increment modifier. That was - I sort of smiled when I saw that because SQRL has something that is the equivalent. And that is, what if, with all of those settings not changed, for some reason you need a different password? That is, this is deterministic, so same stuff in, same thing out. But what if that password, you have to give it to a friend, and now you can't trust it; or the site gets compromised, and they say you must change your password.

Well, the problem is you can't change your password if it's deterministic. But you can change the integer increment to two, and then you get a completely different password because that gets put into the hash, as well, and gives you a whole second, you know, another password. But there again is another problem. Now something has to be remembering for you all the settings you used for every place you visit.

So a lot of people have asked me what I think. Now you know. It's like, okay, that's what it is. Maybe it's the right thing for some people, but I don't think it's the right thing for most of us. And, boy, LessPass? No.

Leo: Yeah. And LastPass does the job. The problem, of course, LastPass has that single point of failure issue, where all of your passwords are stored in one blob on their server.

Steve: And maybe someday we'll be phasing out. We'll see.

Leo: Yeah. Oh, I see a smile on your face. Have you been talking to Joe again?

Steve: So, okay. The confusion of, okay, now we're no longer talking about LessPass; right? We're talking about LastPass. LastPass announced last Wednesday, in fact Joe posted, he says: "I'm thrilled to announce that, starting today, you can use LastPass on any device, anywhere, for free. No matter where you need your passwords - on your desktop, laptop, tablet, or phone - you can rely on LastPass to sync them for you, for free. Anything you save to LastPass on one device is instantly available to you on any other device you use."

Okay. So what happened was people worried what this meant. They said, "Wait a minute, LastPass is now free? Well, then, what's their revenue model?" What are they, you know, I mean, people were immediately suspicious. So here's what happened. LastPass has three plans: the free plan; the premium plan, which is just a dollar a month; and then the enterprise plan, that has all kinds of extra multiuser collaboration features and stuff. Until last Wednesday, mobile devices required the premium plan. So that meant, so for free, you could sync all of your desktop and laptop browsers, but the mobile apps only would run with the premium plan. What they did was they just - they changed that one caveat so that now the non-paid free plan includes mobile devices. So,

yay. That's just a nice benefit and a step forward.

Leo: And they'll still make money. I mean, we pay a lot for the enterprise version. It's well worth it. I continue to pay for a premium version personally.

Steve: As do I.

Leo: Just to support them.

Steve: Yup.

Leo: I think this is doing the right thing. It's encouraging more people to use LastPass. And I hope that that will help their business, as well.

Steve: Well, and it's funny because I, in researching this, I dug back into time and found a Q&A where they asked themselves the question, why does LastPass - now, this is in the past, previously. "Why does LastPass require a premium charge for the mobile apps?" And they answered: "We give away the majority of our features and service for free because we sincerely want to make password management accessible for everyone. We've determined that mobile access and a few other advanced features create added value for our customers, and the premium service provides unlimited access to all of those added features. While we've striven" - and I stumbled over the word "striven." Striven? Really? You want to - I guess that's right.

Leo: Or have strived.

Steve: "While we have striven" - that just seems, you know, although I didn't know what penultimate meant, so I'm not anyone to criticize - "to offer as much as we can for free, our Freemium business model allows us to maintain our service and further its development. We continue to add new features to both the free and premium services and increase the value for our users." So I just think this is a step of maturation for LastPass. And bravo.

Leo: Happy to see it, yeah.

Steve: And I'm glad that they could do it, yeah.

Leo: Yeah. And one thing LastPass does that this other one, LessPass, doesn't do and I find very important is application passwords. So LessPass works great for browser web-based passwords, as does SQRL. But I still want to have, you know, my applications have passwords, too. And I kind of - you don't get the benefit, really, of seeing LastPass at its finest. On Android, LastPass fills in passwords, has all sorts of nice features they can't do on iOS. Because you use an iPhone, I know you don't see

those. But it's really - it's a great solution for everything.

Steve: Yeah. I use it slavishly.

Leo: Yeah, yeah, me, too, yeah. So.

Steve: So. Okay. So a team of university researchers in Germany conducted the first formal security analysis of OAuth 2.0. Okay, now, just step back for a minute. The first formal security analysis, meaning...

Leo: What?

Steve: ...it had never had a formal security analysis. However, we just all started using it. So that people that know what we're talking about, to be sure, OAuth is that increasingly ubiquitous "Sign-in with Facebook," "Sign-in with Twitter," "Sign-in with Google" that we're all seeing increasingly everywhere. What it boasts is convenience because, rather than needing to create new credentials for a site where you don't already have an account, you can just sort of defer them, delegate them to a well-known site where you do have an account - Facebook, Twitter, Google and so forth.

I've never been a fan. We've talked about it extensively in the past. The [audio dropout] is exceedingly complex, which is always a problem for, as we know, complexity is the enemy of security. It's always felt like an inherently error-prone kludge because your web browser bounces around among sites in order to pull this off. And it's also annoying, or users should be aware, that it is a three-party solution, which inherently leaks privacy information to the identity provider, that is, to the Facebook or the Twitter or the Google that you're signing in through because they know by definition who you are because you're providing your Facebook, Twitter, or Google credentials to them. And they know everywhere you log in on the web using them with this method because they are involved in bouncing your browser back to where you want to authenticate after you've given them your credentials.

So you betcha people like Facebook, Twitter, and Google are all excited to offer OAuth sign-in because it is allowing them to track you by a known account on their service everywhere else you use them to sign in. So obviously that's not going to make me a fan. And it's a three-party solution which is inherently a concern because it's not just between you and the site you're visiting.

Okay. So this group tackled a seriously hairy problem because, again, a formal analysis of two plus two is not difficult. A formal analysis of OAuth with the bizarre level of special cases and complexities, and there's four different ways to do some things - I'm not kidding, four ways - to unwind that and tackle it, when they say a "formal security analysis," they don't mean, yeah, we ordered pizza and discussed it and decided it was good. They mean with mathematical rigor, which is incredibly difficult to reduce OAuth to. In fact, it required a 95-page technical report which we don't have time in this entire life of the podcast to do, so we're not going to.

But I'll just - there was a 12-page summary. And just to give you a sense for it and for what they wrote: "The OAuth 2.0 protocol is one of the most widely deployed

authorization single sign-on protocols and also serves as the foundation for the new single sign-on standard OpenID Connect." And just to pause for a second, it's like, yes, because it's simple. And unfortunately this is a classic case of ease of use and simplicity completely trumping security and privacy. But, you know, this is still the Wild West.

Anyway, continuing, they said: "Despite the popularity of OAuth, so far analysis efforts were mostly targeted at finding bugs in specific implementations" - and, oh, boy, our next piece here is the mother lode of that - "and were based on formal models which abstract from many web features or did not provide a formal treatment at all." That is, there never was any formal foundation laid. It was just like, this information bounces here, and then it gets sent over there, and so forth.

"In this paper," they write, "we carry out the first extensive formal analysis of the OAuth 2.0 standard in an expressive web model. Our analysis aims at establishing strong authorization, authentication, and session integrity guarantees, for which we provide formal definitions." Again, you need that if you're going to - you need formal conclusions and then demonstrate that this protocol brings them, yields them, results in those.

"Our modeling and analysis," they write, "of the OAuth 2.0 standard assumes that security recommendations and best practices are followed in order to avoid obvious and known attacks." So that is to say, they're assuming that the implementation didn't make any mistakes, and that the best practices that are understood for OAuth were followed. "When proving the security of OAuth in our model, we discovered four attacks which break the security of OAuth. The vulnerabilities can be exploited in practice and are present also in OpenID Connect.

"We propose fixes for the identified vulnerabilities and then, for the first time, actually prove the security of OAuth in an expressive web model."

Leo: Wow.

Steve: Yes. "In particular, we show that the fixed version of OAuth, with security recommendations and best practices in place" - that is, assuming no other mistakes were made - "provides the authorization, authentication, and session integrity properties we specify. The problems were reported to the OAuth and OpenID Connect working groups who confirmed the attacks." So again, just a beautiful piece of very difficult work because the OAuth spec just is a mess and a kludge.

Leo: Where did it come from? Was there an OAuth working group? Or did it come out of Google?

Steve: Boy, you know, we covered it years ago. I think it began...

Leo: Isn't there an OAuth 2.0, like isn't there a...

Steve: Well, OAuth 2.0 is where we are now.

Leo: Oh, we're looking at OAuth 2.0, then.

Steve: Yeah, yeah.

Leo: Oh, okay.

Steve: Yeah, this is 2.0. So this is the second iteration. And the idea was clever, that is, I mean, you know, essentially it's I want to authenticate to a site that doesn't know me. But I want to use a site that does know me. Well, okay. And so a group of people said let's do that. And so they came up with a way of essentially using browser redirection to send you to a site that knows you, where you authenticate, and then that site contacts the server of where you want to be known and sends your credentials there, and then bounces your browser back to there.

So the user's experience is textbook perfect. You know, log in with Facebook. Everybody who's on Facebook, it's like, oh, yeah, why not? Well, you know. And Facebook, again, they're seeing everywhere you log in through them as a consequence of this. But again, no username and password needed. Login with something you already have. So it's a huge appeal. But it's not without its problems.

Leo: Well, should we stop - should I stop using it? I mean, I use it all the time.

Steve: No. I wouldn't say so. I would say - the only thing I would say is that somebody who is a high-risk target could be attacked by using that. And everyone should understand that who you're logging in with is definitely logging what you're doing. They're accumulating that data for their own purposes. There's no question about it, that that is part of their vacuuming, which unfortunately we know all these big companies do now. So again, if that doesn't put you off, fine. And if you're not a high-value target, fine. But I think it's worth understanding that you're making a tradeoff for that convenience. There is some privacy tradeoff. And that is all, after these four problems have been fixed, and there's no implementation error.

But a trio of Chinese researchers from the Chinese University of Hong Kong, which actually is right next door to the Post Office, they have examined 600 Android apps. Now, they're picking on Android only because Android apps are easy to examine. But they are sure the same problem exists in iOS. What happened is that OAuth was designed for browsers. And it makes some assumptions about the way the browser works with this bouncing around. But again, because it is an appealing hook, you know, it's an appealing offer, apps are now allowing you to sign into them with OAuth, you know, sign into this app with your Facebook account.

Well, it turns out that apps did not implement the OAuth protocol correctly. They examined 600 of the most popular U.S. and Chinese Android apps. In 41 percent of the 182 apps which supported OAuth single sign-on, they found significant problems with the implementation. So here the problem was not the spec. That was also broken, which we just talked about, that the German researchers have now found and fixed, or at least specified the fix for, and it'll get deployed. This is implementation mistakes which apps have made.

The problem is that many iOS and Android apps offering OAuth have failed to rigorously follow the standard. When testing real-world implementations, the researchers found that the developers of a huge number of these Android apps were not properly checking the validity of the information sent from the ID provider, which, for example, in this case would be Facebook or Google. And there's a major provider, SINA, S-I-N-A, that is a Chinese company.

The vulnerabilities resided in the way many app developers implemented OAuth. When a user logs in via OAuth, the app should check the ID provider, like Facebook, Google, or this Chinese firm SINA, that they have the correct authentication for those sites. If so, OAuth provides an access token from the ID provider's server, which is used with the server of the mobile app, that is, the site that the mobile app is a frontend to. So this allows the apps server to gather the user's authentication information and then let them begin logging in with their Facebook and Google credentials.

But the researchers found that, for many Android apps, again, 41 percent of those that offer OAuth sign-in - and again, we believe iOS has probably the same problems because no one is enforcing or testing or checking except here's the first instance of it. The app developers did not bother to verify the validity of the information returned by the ID provider. They failed to verify the signature attached to the authentication information retrieved from Facebook and Google. That is, part of the OAuth spec says "sign this so it cannot be changed." So, yes, Google and Facebook and SINA are signing their provider packet that goes back to the app. The apps don't check the signature. So what that means is they won't notice if that gets changed.

And then, in a different case, the app server would only look at the return's userID and login the individual without checking the attached OAuth information to see if they were linked. So what this essentially amounts to is the appearance of logging in with none of the actual safeguards implemented, which the protocol specifies and requires. It works. But it doesn't work in any way securely. So the upshot of this is it's possible for a remote hacker to download the vulnerable app, login with their own information, that is, log into Facebook or Google with their own information, but change the username that comes back to them to that of the target individual, and then submit that to the app. They will be logged in as that user on the app.

So it's a complete subversion of OAuth; 41% of Android apps that were looked at out of the 180-some that do support OAuth, out of a set of 600 that these guys looked at, had this problem. That is, the appearance of it all working, but none of the safeguards that the OAuth spec explicitly requires to prevent exactly this problem.

Leo: Geez.

Steve: Yeah. So in their paper they suggest that part of the problems are that the developer document is not very clear. OAuth was originally designed for websites, not for apps. And there are some implementation errors by some identity providers, that is, the likes of Google or Facebook, probably not those guys, but some of them have their own problems. And so as a consequence, to get stuff to work, the apps may have backed off on some of the rigor for the sake of compatibility, when in fact in doing so they completely sacrificed security. Wow.

I got a couple notes from people, from our listeners in the U.K. And I just thought this was sort of interesting. I got a kick out of the timeline. A major automobile insurer named Admiral announced that they planned to use data from public postings on

Facebook to decide how to rate customers. They called it "firstcarquote," and on their page it says, "How do I get a quote?" And then they say, "The firstcarquote online app is really easy to use. Enter car registration. Agree to our terms and conditions. Connect with Facebook. Answer 10 simple questions. Get a quote. Call us to buy your coverage."

So following this announcement, the Guardian on the 1st, so one week ago on Tuesday, had a story announcing this, said: "Admiral to price car insurance based on Facebook posts." The reason I'm sharing this is that some of the details are just too fun. And then their sub-headline was "Insurer's algorithm analyzes social media usage to identify safe drivers in unprecedented use of customer data."

And they wrote: "One of the biggest insurance companies in Britain is to use social media to analyze the personalities" - I mean, the language is just so creepy - "to analyze the personalities of car owners and set the price of their insurance. The unprecedented move highlights the start of a new era for how companies use online personal data and will start" - I hope - "a debate about privacy," they write. "Admiral Insurance will analyze the Facebook accounts of first-time car owners to look for personality traits that are linked to safe driving." Okay. Get out your AI.

Anyway, "For example," they write, "individuals who are identified as conscientious and well-organized will score well. The insurer will examine posts and likes by the Facebook user, although not photos, looking for habits that research shows are linked to these traits. These include writing in short concrete sentences" - I think I would fail that one - "using lists, and arranging to meet friends at a set time and place, rather than just 'tonight.'"

Leo: 4:37 p.m. on Thursday the 19th.

Steve: Yes, rather than just "tonight." Oh, lord. In contrast, evidence that the Facebook user might be overconfident, such as the use of exclamation marks and the frequent use of 'always' or 'never' rather than 'maybe,' will count against them.

Leo: Wow.

Steve: Wow is right.

Leo: You know, though, I don't have a problem with that if you opt in, and there's people who don't do it. Maybe you get a much better deal; right? Maybe it's so effective.

Steve: Well, and that was the hook, yes. But it took 24 hours, and then The Verge reported. Their headline a day later, on November 2nd: "Facebook blocks insurer exploiting user data to find 'conscientious' drivers."

Leo: Oh, well.

Steve: With the subhead: "Admiral Insurance wanted to analyze Facebook users' posts

to see if they would make good drivers." And then The Verge wrote: "Facebook has blocked one of the UK's biggest insurers from using the social media network's user data to set insurance rates. A recently launched scheme" - I got a kick out of the word "scheme." It seems kind of skeezy. The scheme "from Admiral Insurance targeted first-time car owners, offering to analyze their Facebook posts to see if their personality traits matched those of successful drivers. Participants were told they could save as much as 350" - which is currently $429 - "a year on their car insurance if they were judged to be conscientious" as well as organized.

"The scheme, named firstcarquote," wrote The Verge, "was set to launch this week; but, as first noted by U.K. privacy advocates Open Rights Group, the app has been blocked by Facebook from accessing user data. The initiative from Admiral Insurance would contravene Facebook's Platform Policy, which states that developers cannot 'use data obtained from Facebook to make decisions about eligibility, including whether to approve or reject an application, or how much interest to charge on a loan."

Leo: Isn't that interesting. Huh.

Steve: Yeah. So Facebook was ahead, yes, Facebook was ahead of the game on this one and nipped them in the bud. So, wow. But, you know, just the...

Leo: You know, I get reduced insurance because I don't smoke. But they don't check my Facebook postings to find out if I smoke. They simply ask me. Yeah, I have mixed feelings about it. But you're right, I mean, the fear would be that people would be denied insurance because there's a picture of them eating a doughnut, which I might have just done.

Steve: Well, it's going to be - we are navigating some new waters because, as we all know, there is, you know, for us old-timers, this seems a little George Orwellian. But apparently the studies have shown that the Gen-Xers, or are we on Z now, maybe, Y or Z, that, yeah, it's like they just don't care. Like, okay.

Leo: Wow.

Steve: So I forgot to mention at the top of the show - our listeners are going to love this - we are about to, in two stories, to have a user-side action. It turns out we can all determine within some limits whether our ISPs are blocking outbound spoofed packets from our own network. But first, GCHQ in the U.K. has announced that they want Internet providers to rewrite systems to block hackers. Now, I consider this premature, but encouraging. So of course this is following on the massive outage a couple of weeks ago at Dyn, launched by the Mirai botnet and just the general problem of DDoS attacks. And in fact, Leo, if you want to put up on the screen, there's a graphic here at the bottom of the story, shows the daunting number of attacks that are currently occurring.

So the GCHQ "is urging Internet providers to change longstanding protocols to stop computers from being used to set off large-scale cyberattacks. The government's cyber-defense arm said it plans to work with networks such as BT (British Telecom) and Virgin Media to rewrite Internet standards to restrict spoofing, a technique that allows hackers to impersonate other computers and manipulate them to carry out anonymous attacks.

"Ian Levy, technical director of GCHQ's National Cyber Security Centre, told the Sunday Telegraph: 'We think we can get to a point where we can say a U.K. machine can't participate in a DDoS attack. We think that we can fix the underpinning infrastructure of the Internet through implementation changes with ISPs and [what they call] CSPs, Communications Service Providers.'

"The plan would involve changes to the Border Gateway Protocol (BGP) and Signaling System 7 (SS7)" - that we've been talking about recently because it doesn't provide authentication natively and thus is prone to attack, as we know - "standards that have been in place for decades and are widely used for routing traffic. GCHQ wants providers to stop the trivial re-routing of U.K. traffic and help prevent text message scams. The Internet Service Providers Association (ISPA), the body that represents ISPs, expressed skepticism, saying GCHQ was applying a 'we can fix it, it's easy' approach to a complex, historic system."

And of course we know, because we've been discussing this a lot, that spoofed attacks are only one class of DDoS. And back in the day when an individual probably had valuable bots and not an infinite supply of them, spoofing was a way to keep them hidden. And it was difficult to protect against. But in this day and age, where every light bulb is able to attack, and there's apparently tens of millions in the Mirai botnet, who cares if it's a known IP?

And so what they're talking about, of course, is egress filtering. Sometimes it's referred to as ingress, if you're ingressing to the public Internet, but you're egressing from a private network. And we've often said this is just not a problem. Well, our next story looks at that. But, for example, here, in just one week, October 1st through the 7th, so last month, first week of last month of October, the number one attacked country was the U.S., where 256,212 DDoS attacks occurred. Second, and not many fewer, was Russia, 211, almost 212,000 in Russia. Then France at 118.6 thousand, India at 34.8 thousand, and Germany at 30.7 thousand.

So, you know, DDoS attacks are happening. And in some of the commentary that I didn't putting here in the show notes, people have criticized, exactly as we have, noting that, okay, so U.K. ISPs won't be able to host spoofing bots, but everybody else will because, as we know, the only way to kill all spoofing is for all providers to prevent spoofed packets from leaving their own control. That is, it's at that point where the packet is leaving an ISP that it can be inspected to see if it has a legitimate return address. And it's worth noting, too, that you could spoof within the ISP, and then it might not detect it.

So if you had an ISP, for example, Level 3 - I was going to say Level 4, but it's a 4-dot. Level 3 has the 4-dot space, 4-dot everything. Well, so if they were only checking the packets as they finally went out on the Internet to make sure they began with 4-dot, you could still have 24 bits, the remaining 24 bits of the 32-bit IP randomized, and it would pass right out through the egress filter. And that's one of, what, 16 million addresses available. So you still get to hide, even if egress filtering is in place. So I think this is hopeful because the size and the severity and the concern over these attacks is raising the level of interest, and people are beginning to talk about getting this thing fixed, which is good.

I then came across a story, and I'd forgotten about these people, Leo. We talked about them years ago, maybe a decade ago, CAIDA. I first encountered them back in the early days of DDoS attacks when DDoS was still novel, because what these guys had done, what they'd done is very clever. They had arranged to borrow huge chunks of the then-unallocated IPv4 space. And as we know, 10 years ago half of the IPv4 space - doesn't seem that long ago, but the growth has been rapid. Half of the IPv4 space was still

unallocated. It was, you know, universities had huge unused chunks. And famously - boy, I'm blanking on a name, our favorite ad hoc network, I want to say - oh, Jungle Disk. They were using 5-dot for their network because nobody had ever used any 5-dot. And so that's the way they avoided colliding with anybody else in the...

Leo: Hamachi. Hamachi.

Steve: Oh, you're right, Hamachi. Thank you, yes, not Jungle Disk. Jungle Disk is network backup.

Leo: Right.

Steve: Hamachi.

Leo: Hamachi.

Steve: Thank you. Hamachi. And so...

Leo: It was a really clever kind of a hack, actually.

Steve: It was great, yes. And so what this Center for Applied Internet Data Analysis, CAIDA.org, what they did was they arranged to have these unallocated chunks routed to them. And the idea was that spoofing software at the time was just using a pseudorandom number generator for its source IP. So a percentage, and a relatively high percentage because a lot of unallocated space was available, a percentage of the attacking packets would have the unallocated space as its source IP, meaning that when, if a SYN packet was sent to a server to spoof a TCP connection, the SYN/ACK response would go into that unallocated space, which this Center for Applied Internet Data Analysis had borrowed. And since it had never been used in the history of the Internet, there was no valid reason for any packets ever to enter. So that meant that those that did were an indication of DDoS attacks.

So this was a decade ago, the first, as far as I know, or at least the most comprehensive DDoS monitoring system. And so they began to provide statistics for where attacks, like they also knew what the attacks were against because when an attack bounced off of a valid server, the source IP of the incoming packet to the unallocated space would be the legitimate IP of the victim being attacked. So neat work that they were doing.

Well, they've moved on. And in fact they wrote: "Seeking to minimize Internet susceptibility to spoofed DDoS attacks" - and Leo, if you want to go there, it's CAIDA.org. Under the top-level menu is Projects, and then look for Spoofer underneath the Projects at the top level of that site. Then they write: Seeking to minimize Internet susceptibility to spoof DDoS attacks, we are developing and supporting open source software tools to assess and report on the deployment of [what they call] Source Address Validation (SAV) best anti-spoofing practices. This project includes applied research, software development, new data analytics, systems integration, operations and maintenance, and an interactive analysis and reporting service.

"We have developed and support a new client-server system for" - and this is why I want our listeners' ears to perk up - "Windows, Mac OS, and Unix-like systems that periodically tests a network's ability to both send and receive packets with forged source IP addresses" - in other words, spoofed packets. "We are in the process of producing reports and visualizations that will inform operators, response teams, and policy analysts. The system measures different types of forged addresses, including private and neighboring addresses." Oh, and in fact neighboring spoofing is what I was just referring to. "The test results will allow us to analyze characteristics of networks deploying source address validation," in other words, network location, business type and so forth.

So they have downloadable, free, open source executables for Windows, Mac OS, Ubuntu; and the source is also available. I ran it on my Win7 machine this morning, and it is very cool. I mean, this is too much fun. Our listeners are going to go crazy for this.

Leo: I'm installing it now. It's safe; right?

Steve: Oh, absolutely.

Leo: Okay.

Steve: Yeah, yeah, yeah. It maybe took about 10 or 15 minutes to run. There is a button at the bottom that opens the log. You definitely want to do that, just because then you can see what it's doing. When it was all finished, it reported that over IPv4 it had identified my carrier of my cable modem, which is Cox, as ASN 22773. And then it said below, "Spoofed private addresses: blocked. Spoofed routable addresses: blocked." But then it also gave me this really cool link, which was to their site, with a big long serial number after it - and there it is running on your system, Leo - a big serial number after it. You click that.

Actually, I don't think it was clickable, so I copied and pasted it into the URL. What I got was this wonderful graphic map built from my test which showed red where spoofed packets were transiting. And in my case it showed the host machine and my router, and it said the comment: "Your host is behind a NAT router or firewall which rewrites the source addresses of the test traffic. To test your provider's network further, you must remove the NAT router firewall and connect directly." Well, needless to say, I'm not doing that.

And this comes right back to the question, I mean, sort of the puzzle I was asking us rhetorically last week, which was how can spoofed packets even egress through a NAT router? Why would a NAT router inherently, that is going to be doing packet rewriting, why wouldn't it drop them? Well, pfSense, we now know, does, the one I'm using. I can't wait to hear from our listeners whether the Ubiquiti EdgeRouter X does also, in which case you're not able, your network cannot send spoofed packets out. They are dropped at your own border.

Leo: Now, if I have an Astaro gateway, which we do - we run the Sophos, you know, they bought the Astaro company - does that impact this? I mean, is my own security part of this? Or no?

Steve: No, no.

Leo: So it's really just testing the external network.

Steve: Well, yes. This is a public service that all of our listeners can perform because what we're doing is we're giving this group the data on our providers, is what it amounts to. And unfortunately, because my router is dropping, is blocking any spoofing at its border, I'm not able to provide information to them about Cox.

Leo: So I might not be as useful, either.

Steve: Yes.

Leo: We're on Sonic.net here, so I'm sure...

Steve: And I'm super interested because we've been talking, you know, like Linksys routers, Cisco routers, Netgear routers, Ubiquiti EdgeRouters. I can't wait to learn, and this software will show you, whether your router will pass spoofed packets out. And the diagram is this beautiful network diagram. Wait till it gets done and it gives you a link, Leo, because it'll show - you get this network diagram of where everything went, all sorted and organized. But the upshot of this - so first thing is I would love our listeners who are inclined to do this. And, I mean, I would love to know if Cox is blocking this. And I can set up a secure machine. I'm afraid, obviously, to not have a NAT router between me and the Internet because I depend upon it for firewall services to such an extent.

Leo: Ooh, yeah, yeah, yeah.

Steve: But if there are NAT routers that do not rewrite spoofed packets, I would love to know. But the upshot is they are now beginning to collect data. I hope our listeners will do this because this will help to dramatically expand the knowledge of what providers are blocking. But now look at the chart. I put three charts in the show notes. It's also available in those pages. They've got both lists and graphical presentations of what they have found so far.

In the announced, the so-called announced IP space - oh, and I forgot to mention that my system is still IPv4 only, so the test does run under IPv6. But it just noted that I don't have IPv6 operating in my network, so it didn't do testing there. But, okay, 75.3% is unspoofable. So sort of quietly, without bringing any attention, ISPs have in fact been taking action. And I read some quotes saying, yeah, we don't like DDoS attacks. We don't want spoofing, either.

Leo: No.

Steve: So this has been going on. So they got inconsistent results from 6.9, so call it 7%, and a confirmed spoofability in the balance of 18%. But still, three quarters of the IP

space will not allow spoofed packets out. So what this essentially means is this is already probably to the point where bots won't spoof because spoofing bots will be less effective. Their bullets will hit a wall and not reach the target.

Leo: But as we talked about, nowadays with IoT, spoofing isn't as necessary as it used to be.

Steve: Right. But it's also nice, you know, spoofing is a violation of the Internet protocol. So it'd be nice if we just nipped that in the bud, even though it's not going to give us any affirmative protection against non-spoof attacks, which are all today still very practical. Anyway, I love the idea that there's something our users can do. Did your test finish?

Leo: Still running it. Not quite done yet.

Steve: Okay.

Leo: It went through all the way to the end of the progress bar and now is doing something else. So I guess Test 5 now.

Steve: Oh, it just does all kinds of cool stuff.

Leo: How many tests are there?

Steve: I think it was seven, if I remember right. And it ends up with a whole bunch of traceroutes out to various points. So what it's doing is it's looking at your endpoint to something like 13 other servers that they operate, scattered all over the globe. So it's not just to them.

Leo: Hmm, interesting.

Steve: It is, you know, to all over the place. And so they build a really comprehensive chart, and you'll end up seeing this nice spider diagram of your network as soon as it's finished.

Leo: Neat, neat. I'll show it to you as soon as it comes up.

Steve: Cool. In the meantime, clearly Microsoft is understanding that they've gotten the feedback that Windows 10 updates and this often problematical insistence on restarting is causing their users problems. We of course discussed last week the group policy editing that can be done to prevent Windows from ever downloading updates. It will notify you, but it will not update. It will not download them. The problem is that that's the only choice you have. If you let Windows update them, it ultimately will insist and override any attempts not to.

In Microsoft's own posting, they wrote: "You can use Group Policy settings or mobile device management" - that is for Windows Mobile 10 - "to configure when devices will restart after a Windows 10 update is installed. You can schedule update installation and set policies for restart, configure active hours for when restarts will not occur, or you can do both. After an update is installed, Windows 10 attempts automatic restart outside of active hours. If the restart does not succeed after seven days" - then they say in parens (by default) - "the user will see a notification that restart is required. You can use the 'specify deadline before auto-restart' for update installation policy to change the delay from seven days to a number of days between two and 14." So again, you can't push it any further than two weeks. But really that's hopefully plenty of time.

Anyway, I put the link in the show notes. And that's just a tiny snippet from an extensive posting. That TechNet article at Microsoft.com that I link to in the show notes goes through the whole enchilada, so all of the group policy settings which are available to help people configure Windows 10 updates the way they want them. So I just wanted to share that with everybody.

Leo: I have the graph. And just like you, I'm blocked by my NAT.

Steve: Ah, right, exactly. That's exactly right.

Leo: Preventing rewriting of packets.

Steve: Cool chart, though. Isn't that neat?

Leo: Yeah, that's neat. Are these all nodes, these numbers here?

Steve: Those are all router hops. And so that's the result of the traceroute that was done. So all the endpoints are where it was checking, and it watched the packets hop out and then track where they went. And so, for example, if it turns out that consumer routers - and we'll know a week from now. I'll tell everybody because I'm sure our listeners will try this and send me the results. What you would expect to see is those red arrows propagating further out.

Leo: Right.

Steve: And where they get snubbed would be a point where packets are being checked for validity.

Leo: Right. So I cannot - I'm blocked. I cannot do it. I'll go home, though, and do this on my ASUS router and on my Eero router.

Steve: Ooh, yeah, yeah, yeah, good.

Leo: Yeah, and let you know.

Steve: Nice. Microsoft also announced that they are winding down, they are end-of-lifing their Enhanced Mitigation Experience Toolkit, EMET, E-M-E-T. And I won't go through this in detail. I'll just say that their posting sort of talks about how good it's been for so long; that their rationale for it was that major Windows releases to address problems that required architectural changes were, as we know, spaced many years apart. That wasn't often enough to allow them to evolve Windows in ways that just security patches were insufficient to fix. So they regarded EMET as their stopgap, that is, as an OS adjunct which could evolve much more quickly and live with initially XP and then 7 and then Windows 8 and even 10.

But due to the new architecture of Windows 10, where it is no longer a static OS over the long term, but it's Windows as a service, now Windows 10 itself will be able to evolve. And they explain that there was a fundamental problem, a limitation that EMET wasn't part of the OS. So there were things they wanted to do with it that it couldn't do because it wasn't the OS. It was better than nothing. But what they now have is better than the earlier OSes with EMET. So EMET is winding down - they've, like, formally announced end of life - to be replaced by the evolving Windows 10, which as we know will change as needed moving forward.

Okay. And this is too simple. I'm surprised this flaw existed in Gmail. The good news is it's been found, and I'm sure Google will be on it, if they haven't already. A student in Pakistan, the CEO of a company called Security Fuse, Ahmed Mehtab, discovered a flaw in Gmail's authentication and verification methods when a confirmation email bounces. If a user has more than one email address, Google lets the user link all of the addresses and also lets emails of the primary account be forwarded to secondary accounts.

Mehtab identified an inherent flaw in the verification bypass method adopted by Google for switching and linking email addresses, which lead to the hijacking of email IDs. He discovered that the email addresses became vulnerable to hijacking when one, any one of the following conditions occurs: when the SMTP of the recipient is offline, so the mail bounces; or the email has been deactivated by the recipient so it can't be sent; the recipient doesn't exist at all; or it's an invalid email ID so, again, a send fails; or the recipient does exist, but has blocked the sender, another reason for bounce.

So the attacker tries to verify the ownership status of an email address by emailing Google. Google sends an email to that address for verification. The email address cannot receive the email; and, hence, Google's mail is sent back to the actual sender with the verification code in it. This verification code is then used by the hacker, and the ownership to that particular address will be confirmed. So a startling flaw in essentially an edge case of multiple email addresses, one of which cannot receive the confirmation email. And so, due to the fact that the confirmation email contained the verification data, it can be made to bounce back to the attacker, who then gets the verification data and is able then to verify their ownership of an email account they don't own.

Leo: So don't block any senders.

Steve: Clever and deadly, yeah.

Leo: Yeah.

Steve: I did want to mention, just as a public service announcement - this is now a couple months old. But the very widely used MySQL, you know, MySQL server, had some serious problems discovered back in August. And they were responsibly disclosed by a researcher with Legal Hackers. There are two known exploits. He waited more than 40 days. So both the official Oracle MySQL and its MariaDB and PerconaDB all have these problems. So they existed, not only in MySQL, but in both of those forks. So both of them, MariaDB and Percona, both updated their servers and issued patches. After 40 days, Oracle had not.

So he decided to go public with the details of the zero days. Last Tuesday he released proof-of-concept exploits for two of these vulnerabilities. So Oracle has since patched MySQL. Both the forks, Maria and PerconaDB, have been updated. So attacks are in the wild. I just wanted to make sure that any of our listeners who have MySQL on servers that may need to be updated know about this. So, and I did get the notification on my FreeBSD box because I'm a MariaDB user, and patches are available, and it was easy to do.

And, finally, I loved Paul Thurrott's tweet a couple days ago. He tweeted: "Here we go again: Microsoft's popping up ads from the Windows 10 toolbar." So we get to be the customer AND the product.

Leo: Oh, well.

Steve: Okay.

Leo: Oh, well.

Steve: And a bit of errata. Last week I went on at great length addressing the question of replacing domain names with IP addresses and explained that, well - and this, of course, was brought on through a Q&A where a listener was asking, hey, if DNS goes down, could I just bookmark the IPs of the sites and get there? And I explained the various problems that that would cause, completely forgetting that everything I was saying applied only to non-HTTPS connections, which we now know are becoming increasingly rare. That is, this of course absolutely breaks HTTPS.

Leo: You can't enter a website by number if it's secure?

Steve: No, because the certificate has the domain name.

Leo: Oh, it's the name.

Steve: Yes. So you get a "no name match." Right off the bat, I mean, it's like, sorry. You can't get here. And I also noted that multihomed sites are increasingly used because the

communication from the browser contains the host name in the TCP, the first TCP packet, or rather the first TLS packet, the SNI extension, the Server Name Identification, and that allows the receiving server to enumerate or disambiguate among many sites that it may be hosting at a single IP. And of course that breaks. If you put an IP address into the browser bar, you're not giving it the hostname information that it needs to determine which site in a multihosted IP you want access to. So I absolutely wanted to correct that, that it's like, yes, no longer can IPs be used, unfortunately. They just break too much.

And lastly, I got an interesting question from Norm in Thailand regarding SpinRite drive orientation. He said: "Hi, Steve. I've been searching for an updated answer for this, but I would like to get confirmation on drive orientation when using SpinRite. I'm sure this was covered before, but I cannot find it." He says, "I've been using SpinRite for a long time. Normally I have all drives mounted flat board down. My newer NAS has some problems with one drive, and I'm planning to run SpinRite on all of the drives. I have a new mini computer now for running SpinRite which is under test now and seems to be doing fine. The new NAS has the drives installed vertically on their side.

"So the question is, should newer hard disk drives be done in maintenance mode with the drive in the orientation that it will be installed in? I think on your new computer they are mounted vertical, if I recall." Good memory. That's amazing how he knows that. I guess from a photo that he must have seen, striven to see. "Did you run SpinRite in that same orientation? Just for information, does the drive run upside down make a difference on newer drives?"

Really great question. And remember back - I was talking about this a couple weeks ago, Leo, where the genesis of SpinRite was that a gal that I was dating 30 years ago had her company's accounting information irretrievable on an old-school linear actuator hard drive. It was a big Seagate something or other. And I managed to recover it by lifting one end of the drive, that is, in the axis that the linear actuator moved, thus adding a gravitational acceleration to the heads. And that allowed - that worked against the servo information that caused the heads to then be able to find the tracks. And by successively low-level reformatting the drive nondestructively while inching the drive back down to horizontal, I was able to get the drive to then read flat, which it had stopped being able to do.

Modern drives do not use linear actuators. They are, as far as I know, all rotary actuators. So there's a pivot on the side. The heads have sort of a cantilevered right-angle head mounting. So it actually rotates as the heads move in and out. As you would expect, this is extremely well counterbalanced. That is, the heads moving will create a little bit of rotational torque. There's no way not to have that. But there is no linear vector torque created as there once was. In fact, I remember way back when a previous company, Minicomputer Technology, where I cut my teeth on hard drives, you could do random seek testing. And these things were like washing machines, and the whole thing shook. It was like a badly off-center load in the spin cycle of your washing machine. I mean, it was just...

Leo: It was terrible.

Steve: It was crazy. Oh, my lord. So drives don't do that now because they are carefully rotationally balanced. And what that means is they are orientation-independent. That said, I do - I mean, clearly Norm in Thailand is a perfectionist, which I appreciate. And yes, my drives are vertical. Vertical is fine because the drives, again, are carefully balanced. They have to be in order to be able to seek as well as they do without just

vibrating themselves to pieces. So, but again, perfectionists, yes. I would run SpinRite on a drive with the same orientation it will have when it's in use. So great question, Norm.

Leo: This is going to go away pretty soon, thought; right? We're just all going to be using SSD drives any minute now.

Steve: You know, we've been predicting that for a while. But the problem is the drive technology is so mature and getting more so. We're now heading to shingled drives, which is the next iteration. We're going to see another ridiculous capacity jump.

Leo: Ugh.

Steve: Yeah.

Leo: Yeah. I mean, what is it? What's the biggest now, 8, 10 terabytes? They're big. Big.

Steve: Oh, they're, I mean, yeah. And I think people get them just because they can.

Leo: Capacity.

Steve: It's like, oh.

Leo: Yeah.

Steve: Although I will tell you I made a mistake of I thought - and I'm sure I told you that, with one of my iPads, I thought, okay, I am not going to load this up with memory. Jobs is not going to cheat me.

Leo: Not going to get me.

Steve: So one of my pads is only 16 billion bytes.

Leo: Oh. Yeah?

Steve: I can't survive.

Leo: No.

Steve: It's constantly complaining that it's out of space.

Leo: You need data, yeah.

Steve: I've got to delete things. So I will never - I love my iPads. I am always getting the most memory I can because that's what happens to work.

Leo: And I haven't bought a computer with anything but SSDs in a long time. I just can't even imagine buying a spinning disk. I admit, you know, you want capacity. But I have Drobos and other external stuff for capacity.

Steve: Well, and backup. See, what SSD does have is spontaneous catastrophic failure. That is the way they die. And hard drives tend not to. Because their technology is just kind of old and obvious, they just die in small increments, which we can typically see and fix. But I run across all these stories of SSDs just dying, bang.

Leo: Yeah, yeah.

Steve: So you really do need them backed up.

Leo: Yup.

Steve: Okay. So following up on last week's Windows AtomBomb, with the details, there's a company, enSilo, who has a team leader, Tal Lieberman, who did a beautiful bit of creative engineering. AtomBombing, as he called it, is performed only by using long-established features of every Windows operating system. There's no need to exploit an operating system bug or any vulnerabilities in an application. So there's nothing to patch. Microsoft cannot, at this point, significantly change or remove these features because all the apps, I mean, like in the ecosystem of Windows, depend upon them.

However, it is the case - I'll put in a caveat - that antimalware, including Microsoft's native AV, will likely quickly adopt some augmented heuristics in an effort to detect the use of the technique that Tal has developed. So, okay. So he called it AtomBombing because this uses something known as the Windows Global Atom Table. And it's one of these features that Microsoft put in in 1.0. It's always been there. And it's not really clear what Microsoft was thinking. You can have local atom tables, or there is a global atom table. And all that means is it is essentially - you can think of it as a user-definable dictionary. That is, you can give Windows a blob, a string, typically, to hold. And it gives you back a token by which - and that's called an "atom." And then you refer to that in the future with this token.

So there's one call, an API call, where you say, here's a null-terminated string buffer. Insert this in the global atom table and give me back the token. And so Windows gives you a little 32-bit widget, just it's like a cookie. And then, in the future, you can have Windows look up for you that original string with the cookie. So it's just sort of like a dictionary. Applications can use it locally for their own private dictionary; or, as I said, there is a universally, globally defined atom table that Microsoft must have thought was

useful for something. I'm not sure why or how it's used.

But if there were a system that had multiple processes, then this would be a means of doing interprocess communications, allowing one process to establish atoms that were accessible by another. But as we know, from a security standpoint, this is breaking process isolation boundaries by definition. That is, it's inherently a concern. I would argue it's inherently insecure, not by itself, but when leveraged with other exploits.

So what Tal did was, as he describes it in his own write-up, he just sort of sat down and went through the API, looking for problems. And he ran across global atoms, and he thought, huh. That's interesting. Because, okay, what normally happens with a buffer overrun, we know that a buffer overrun exists when an application accepts attacker-controlled data, and it somehow overruns the buffer. Sometimes they can be made self-executing. But in this world of data execution protection or prevention, many times the data's area, where the data buffer would be, is non-executable. So some tweaking has to happen. That segment of data needs to be made executable.

And so for that the attackers use so-called return-oriented programming, which we've been talking about in the context recently of address space layout randomization that makes ROP more difficult - but, as we know, unfortunately not as impossible as the designers would like. So the idea is that, with return-oriented programming, you are jumping to known executable code to get little bits of work done, just by jumping to the end of a subroutine that already exists in some code that is executable, like in the kernel, for example. And then it returns to you after finishing the subroutine. But you just use the last few instructions to get some work done on your behalf. And by stringing a few of those together, you can create a result. It's difficult to build a whole malware exploit out of little snippets. But the idea is you can get it to do things like flip the non-execute bit for you, which then makes the buffer overrun code executable, and then you jump to it, and it runs. So again, a hybrid attack.

So what Tal worked out were all the details of using the global - okay. What I should say is that the starting point for that is a buffer overrun. That is, the way the attacker gets their data in an innocent victim process space is a flaw in that code, in a web server, for example, where some crazy response is made to a query, and the buffer wasn't big enough to hold it; and it ends up, you know, the attacker's response ends up sitting in data that is the way the attacker gets the data in. What's haunting about the global atom table is this is a non-defective means. That is, this is an officially sanctioned way of getting one process to have its data inserted into another.

So what Tal realized was that the global atom table created a break in process isolation. So he created - and it's on GitHub, by the way, all open source, all documented for anyone who wants to play. He created, again, because it's not based on any defects, it's a problem with Windows. He created a small return-oriented programming script which - so his application, his attacking application first stores the executable code in a global atom. It says to Windows, here's the string. It needs not to have any nulls in it, so it's got to be a little tricky because a null terminates a string. So it's got to be a chunk of code with no zeroes somehow. That would be a bit of a challenge, too.

But anyway, so he did that. Now his code has been accepted by Windows and is sitting in the OS. He then strings together a short chain of return-oriented programming snippets which he injects and causes a victim application to run. And he gives a bunch of examples. And again, they happen to work. So, for example, the VLAN program, VideoLAN, was exploitable. And Paint was exploitable. He just used those as examples, due to no flaw in them, but just the fact that he was able to get his little return-oriented programming to run in that process, to request the global atom that he had loaded into

the OS, which then transferred it into the process. He then copied it into a read/write/execute buffer and ran it, and pulled it off.

And so this was really interesting because, as I said, normally you need to start with a flaw. You're exploiting a flaw in the victim process. Here any qualified - there are some qualifications required. But any qualifying Windows process can be the victim of this attack. It is local, that is, you need an attacker in the same OS that is an attacking process. So it's not remotely exploitable directly. It does need somebody, you know, you need to have a malicious app available to then exploit across process.

But anyway, beautiful piece of work, Tal. So congratulations. And as I said, Microsoft can't fix this. He only used officially sanctioned, highly used APIs. So the only solution would be for malware, unfortunately, to become increasingly heuristic; to, for example, hook these different APIs and try to detect the malicious use of the global atom table for this purpose and then prevent that from happening. So it's not what we would prefer to have. But thanks to his work, a potential exploit has been found and killed.

Leo: Bravo. And now we can get to the election results as the polls start to close.

Steve: I'm there, baby.

Leo: Your timing is very good.

Steve: Polls are closing.

Leo: Actually, we're going to get to Tech News Today. So if you want to avoid any - I was just listening to MSNBC, and they said that candidates often at this time of the night go and do something besides watch the election results because it's too easy to misinterpret. For a while Senator Kerry thought he had beat George W. Bush in the 2004 election, and the people on his plane were calling him President Kerry, and the people in Bush's plane, George W. Bush went down the aisle thanking them for their hard work and saying, well, we did our best. And of course that's not what happened.

And then, kind of surprisingly, Chris Matthews, who started his political career before he was a reporter working with Tip O'Neill, said that - I can't remember who it was - I think it was Kennedy went to an adult theater. They couldn't get into a regular theater, so they went during the West Virginia primary...

Steve: That'll take your mind off the election results.

Leo: They went to an adult theater in 1960 to avoid finding - hearing early an incorrect result. So I'm not going to say anything, and maybe we should just pretend nothing's happening for the next few hours as polls now start to close across the country.

I hope you've enjoyed Security Now!. Don't forget, we have a wonderful Ugly

Christmas sweater design. We're selling it at Teespring.com/twitmas. And you only have till November 23rd. So just a couple of weeks left, if you want to get that in time for the holidays. The Ugly Christmas - we also have tank tops, Ugly Christmas T-shirts, and Ugly Christmas sweatshirts. "Merry TWiTMAS," they say. And you can get it for a song, well, a song and some dollars at Teespring.com/twitmas.

This show is every Tuesday, 1:30 p.m. Pacific, 4:30 Eastern. And that is 19:30 UTC. You can watch live if you wish at TWiT.tv and/or be in the chatroom at IRC.twit.tv. But I suspect many of you will want to get a copy and save it to your hard drive. We have on-demand video and audio. Steve's got 64KB audio plus transcripts at his site, GRC.com. By the way, while you're there, don't forget to pick up a copy of SpinRite, if you don't already have one, the world's finest hard drive maintenance and recovery utility. And also, of course, lots of other freebies. You can see SQRL, Perfect Paper Passwords, ShieldsUP!, and on and on and on. GRC.com.

Steve: And I'm going to quickly put up the link to the show notes on the Security Now! page at GRC.

Leo: Oh, good. It's not there yet.

Steve: Normally I wait until Elaine has finished the transcript, so I just do it all at once. But this podcast had so many links in it that I want to make them available to everybody. So GRC.com/sn. That'll take you to the Security Now! page. And right at the top you will see the links for 585, only with the show notes at this point, until a day or two, until Elaine has the transcript ready.

Leo: Good.

Steve: And probably a Q&A next week. We will find out what our listeners are thinking and get their thoughts and answer some questions.

Leo: You can ask Steve at GRC.com/feedback, or his Twitter handle is @SGgrc. He takes direct messages from everybody. Ask your questions; we'll get to those next week. We have audio and video at our site, TWiT.tv/sn. And of course you can subscribe wherever you get your podcasts. That way you'll never miss an episode. Add to your collection. Collect all 585.

Steve: Drive yourself crazy.

Leo: Yeah. Listen again and again. Thank you, Steve.

Steve: Thank you, my friend.

Leo: And we will see you next week on Security Now!.