Tor 2.0 with Hidden Services

Description: After catching up with this week's Java vulnerabilities and emergency updates, Steve and Leo examine the recent evolution of the public and free "Tor" Internet anonymizing network. They look at the network's updated operation and its new ability to offer "hidden services" in addition to hiding the identity and location of the services' users.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-394.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-394-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here with an update on The Onion Router. First of all, Tor doesn't even stand for The Onion Router anymore. And now, with hidden services, it's about twice as good as ever before. Tor 2.0, our topic next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 394, recorded March 6th, 2013: Tor 2.0.

It's time for Security Now!, the show that protects you, your loved ones online. And we thank this guy for making it possible, the Explainer in Chief, Steve Gibson. Let's get ready to lock it down. Hey, Steve.

Steve Gibson: That's me. Hey, Leo.

Leo: And you're wearing your Atari shirt. Shoot, I should have worn my Atari cap, yeah.

Steve: Oh, of course, yeah.

Leo: I just was - in fact, I should show you this. There's a fun Macintosh app.

Steve: You took your green hat off.

Leo: Well, St. Patrick's Day is not for another 11 days. There's a really fun app on the Mac that you would really - you and a few old farts like us would appreciate. It's a terminal app that emulates an Atari or a Commodore. It's called a Cathode because you can choose from different cathode tubes. And it has all of the things that we used to hate, like the jitter, the rolling. You have a...

Steve: No kidding.

Leo: There's a degauss button, so it goes [sound] and bounces. So it's really, I mean, it's a real trip down memory lane. Unfortunately, I mean, it's a good way to get a headache. So I don't, you know, it's $10. I'm not sure what the - but it's the actual term, it's the Mac Term app, built into it. So you could, I mean, you can actually use it as a terminal, if you want.

Steve: Ah. So the point is that it's the visual display emulating an old-school CRT.

Leo: Right, right. But if I type "LS" or whatever, it actually works.

Steve: Nice. Nice little fadeout on the cursor.

Leo: Yeah, you see? Oh, you're watching, yeah.

Steve: Yeah.

Leo: And you can change the baud rate. You can - here's a 286. Here's a C64. Let me see. I wonder if they have an Atari. I'm pretty sure they do. Yeah, this might be it. I'll make it bigger. And then [laughing]...

Steve: That was very nice. I don't know that it's worth $10.

Leo: It's not worth $10, but you can't resist it. And then you can try to - here's a 1976 TV. Here's a 1991 TV, with a little bit better - we don't have the scanning anymore. You can even change, I don't know if you can see this, but there's a reflection of a room on it. And you can upload a picture of anything you want so that you can actually see yourself in the thing. Isn't that hysterical?

Steve: They really went overboard.

Leo: It's crazy, man. It's crazy. You can [laughing]... I can't imagine what the purpose of it is. But I just thought you of all people would appreciate it. It is $10. It's on the app store on the Macintosh...

Steve: If you want it, it's on the Internet.

Leo: It's the best reason to buy a Mac [laughing]. All right. What are we going to do today?

Steve: We're going to talk finally about the new, relatively new features of Tor. And interestingly, Tor, we did a podcast on it years ago [SN-070] where TOR was considered an acronym for The Onion Router.

Leo: Right.

Steve: And it's no longer officially an acronym, and it's not all uppercase. I actually am still in the habit of doing that. So my notes have it in uppercase, although elsewhere here I remembered that it should be uppercase "T," lowercase "o" and "r." We've touched on this a few times recently because the topic has come up. And because we've never talked about the recent changes in Tor, actually Tor had its 10th birthday last September. So it's now 10 years old.

But the way it works today is completely different from the way it worked when we first described it. So that's why I called the episode "Tor 2.0," and actually some of their documents do the same thing, because they changed the way the protocol functions. And one of the really interesting new things that Tor offers is sort of the flipside of its traditional role. Traditionally, users used the Tor system to hide themselves from Internet servers that are on the public Internet. So that, for example, if some government or other agency was monitoring a service that was controversial, whether it be political or pornography or just whatever, if you just use a normal connection, they, as we all know, get your IP address.

Well, instead, the original Tor system, as does the new one, allows you to essentially bounce around through and among Tor nodes until finally you emerge from one having sufficiently confused the authorities that they have no idea who you are. So the IP address of the service that sees you is that of a so-called "Tor exit node." And we've talked about exit nodes before and how, in fact, we covered a story a few months ago where some guy was running a whole bunch of Tor exit nodes in his living room, and he got in trouble with the authorities because they saw his IP doing controversial things when in fact he was just an exit node on the Tor network, and it was other people whose identities were hidden.

So this recently came up because we were talking about TorMail.org that offers - that is a user of the newer facility, which is sort of the other side of this, where service providers, like websites or instant messaging or email, can use the Tor system to hide themselves. So Tor users...

Leo: Or, more to the point, hide the person using the service.

Steve: No. No, that's it. Hide the service.

Leo: Oh, hide the service.

Steve: Yeah. See, Tor has always hidden the person using the service. But the services were out on the public Internet. What's now been added is a means for the services themselves to be hidden. So it's possible to offer services that are accessible through Tor, but not accessible on the public Internet. So that no one can see who is using these services, or that they even exist. Or, if they do exist, there's no way to figure out where they actually are. So it's, yeah, it's like the flipside of what Tor has traditionally done. And we're going to - I'm going to explain this week how that's done because...

Leo: This is good because I need reeducation. And as you could tell from last week, I've misunderstood the capabilities of Tor.

Steve: Yeah. So it's cool. And of course we've got our regular Java update emergency happening.

Leo: I can't believe this again.

Steve: Again. We have not gone a week without one, Leo. It's now three of these in the last month. So I think for the last three weeks it's been there's a new emergency update patch of Java, and you need to deal with it.

Leo: I think that really I've figured this strategy out now. If they have a new one every week, pretty soon people figure it's all the same one. Oh, we already heard about that one. And it'll just - it'll say, well, there's just one problem that just keeps, you know, people keep reporting. But this is not. This is a new one.

Steve: No. And in this case it doesn't even fix the problems they know about. They still know about vulnerabilities that have not been fixed just with this latest one. So the latest one patches both the older Java 6 to Update 43, and they had intended to stop moving Java 6 forward, but this is so bad they can't. And Java 6 users are having this exploited against them. So Oracle had no choice but to move it forward. And so they fixed two vulnerabilities, one of which was being used to install the so-called "McRAT" - M-c-R-A-T. And as our listeners know, RAT is now the acronym, sort of the generally accepted acronym for Remote Access Trojan. So, you know, aka botnet node, essentially.

But get this, Leo. They knew about this, which is why they were able to do it so quickly. They knew about this weeks ago, many, and were deciding, well, since it isn't being publicly exploited yet, we're going to hold off and roll that into our planned April update, where they've got a whole bunch of other flaws that they're not doing anything with because no one apparently knows about them yet.

Leo: No need to call attention to this.

Steve: Meanwhile, meanwhile our friend Adam Gowdiak, who has got to be Oracle's No.

1 nemesis, he's the security researcher who keeps looking at Java, and every time he looks at it he finds more things that are really bad and wrong. They came back to him with what he called his "Issue No. 54" last week and said this is not a problem.

Leo: Oh, good. Oh, what a relief.

Steve: We don't agree with you that this is a vulnerability.

Leo: Oh, well, if Oracle says it's not a vulnerability...

Steve: So he said, you know, but Adam knows what he's talking about. I mean, he knows Java better than they do. So he looked at the documentation, and the documentation for Java clearly stated that what they were doing now is not what the documentation says. So his response to them was, well, you either need to change the documentation or change the code. And, oh, by the way, since you made me go back and look at it, I found five more new problems.

[Laughter]

Leo: You made me do it.

Steve: I'm not kidding. He found five more problems because they said, no, 54 isn't a problem. So he said, well, yes, it is, and here's five more. And those are all present in the latest update.

Leo: Now, is he publishing code, or explaining how it's done? Or is he holding onto it?

Steve: No, he's being a hundred percent responsible. He's sending them proof-of-concept code.

Leo: But just to keep them honest he's letting us know he's doing it.

Steve: Yeah, exactly. And to sort of keep the pressure on because we know. I mean, their own behavior says, unless they have to update it, they won't. Even when they know their problems. So the consequence of this, I mean, remember that when we talk about exploits in the wild, those are successful installations of a trojan on people's systems. So it's easy to view this in the abstract, where it's like, oh, yeah, that's happening to other people, so, eh. But it is happening to other people, meaning that Oracle's decision not to patch this when they know about it is resulting in people's systems being occupied with trojan malware because they have Java on their system.

And I think it was Brian who wrote of this. No, maybe it was Computerworld. I think it was - because I did a bunch of just sort of backgrounding to see what people were saying. They made the interesting point that Java was really never intended to be a

consumer platform. It was really more of a corporate platform for corporations to roll out multiplatform solutions, very much, for example, the way the Eclipse IDE that I've been using when I was writing code for the microcontroller, the little TI microcontroller that I was experimenting with, will be again here shortly as soon as I get GRC's new server squared away, it's all Java based.

So there's a useful, I would say important place for Java where software needs to be cross-platform. It's a useful cross-platform solution. The alternative is to write in cross-platform C or C++, but then you need to produce separate binaries that run on each OS, and there's really no good cross-platform solution. Then you're maintaining essentially separate source forks for every platform. Whereas Java, because it represents a unifying virtual machine where the platform differences are subsumed by it, you really can have a single Java blob that will run on all these systems.

The mistake people made is in making it a browser thing. And that's why our advice is take it out of your browser. Get rid of the Java plugin. I mean, really, Oracle ought to just remove it. It ought to be something you manually install if you have a known need for Java. It ought not install itself. So it ought to just, with the next update, it ought to remove it from the browser and then go back to being what it was supposed to be. And then, for people whose companies have browser-based Java, then manually put it in. Instead, they brag about it being in three billion devices. Every time, every week that you get an update, it reminds you, oh, look, we're at three billion places. Yes, and unwanted in most of them. Lord.

Leo: [Laughing] It really is hard to believe. It's unbelievable.

Steve: It's just, hey, we're going through a rough patch in the industry, Leo. I don't know how long...

Leo: Well, I wonder if they want to kill, I mean, if Oracle just says let's...

Steve: This has got to be publicity that is damaging for them.

Leo: It's not good for Oracle.

Steve: And it's all because it's browser based. And Java on the desktop is fine. Just not in the browser. They can't get it right.

Apple did a nice thing, too. They updated Safari a couple days ago to block all but the most recent version of Flash. So people who have not been keeping current with Flash, because Safari's still - they're lagging a little bit behind at Apple, lagging a little bit behind in what everybody else has done, which is taking proactive responsibility for making sure that all these plugins are current. I mean, I don't blame them. We know how Apple feels about Flash in general. So it's no longer being installed by default. They'll run it on your desktop. They will not run it on any of their iOS devices.

But Firefox and Chrome and now IE are proactively taking responsibility for keeping Flash current and checking to make sure that you're running the most recent version and warning you clearly if you're not. Apple's doing probably what it ought to from its

perspective, which is they are updating Safari to say, okay, this is not the most recent, go get it if you really want it. And then they have made changes that we know of where they'll disable Flash if you're not using it actively. It sort of goes to sleep, and then you need to manually reenable it. So, I mean, I'm impressed with that. That's the right thing...

Leo: That's what they do with Java, as well, yeah.

Steve: Yeah, good.

Leo: Everybody's been talking about Evernote. In fact, I'm just trying to log into my Evernote account because not only did they reset passwords, but they've downloaded new apps on all the platforms requiring - because this is interesting. Well, you're going to get into this. But this is an interesting point. If you installed Evernote on, let's say your iPhone, your iPad, your Android phone, you already have a token, and that token's not invalidated.

Steve: Right.

Leo: So tell us what happened. Fill us in. You're the guy.

Steve: Okay. So the headline was 50 million passwords were reset. It's not that 50 million escaped. But what they posted on their blog I think is - it's a classic example of how to do this right.

Leo: Yeah. That was my supposition. And I wanted to talk to you because they had the words that we've said we always want to see, "hashed" and "salted."

Steve: Yes.

Leo: There was some speculation they were using MD5, but nobody's confirmed that.

Steve: Well, yes. It is MD5.

Leo: It is.

Steve: And that's not the strongest hash, but it's probably strong enough. So what I think they did right, though, not only was their technology right, but their PR...

Leo: Their response was right.

Steve: ...response was correct. They called it a "service-wide" password reset, and they wrote:

"Evernote's Operations & Security team has discovered and blocked suspicious activity on the Evernote network that appears to have been a coordinated attempt to access secure areas of the Evernote service. As a precaution, to protect your data, we have decided to implement a password reset. Please read below for details and instructions.

"In our security investigation, we have found no evidence that any of the content you store in Evernote was accessed, changed, or lost. We also have no evidence that any payment information for Evernote Premium or Evernote Business customers was accessed. The investigation has shown, however, that the individual(s) responsible were able to gain access to Evernote user information, which includes usernames, email addresses associated with Evernote accounts, and encrypted passwords. Even though this information was accessed, the passwords stored by Evernote are protected by one-way encryption. In technical terms, they're hashed and salted.

"While our password encryption measures are robust, we are taking additional steps to ensure that your personal data remains secure. This means that, in an abundance of caution, we are requiring all users to reset their Evernote account passwords. Please create a new password by signing into your Evernote account. After signing in, you'll be prompted to enter your new password. Once you've reset your password on Evernote.com, you will need to enter this new password in other Evernote apps that you use. We are also releasing updates," as you said, Leo, "to several of our apps to make the password change process easier, so please check for updates over the next several hours.

"As recent events with other large services have demonstrated, this type of activity is becoming more common. We take our responsibility to keep your data safe very seriously, and we're constantly enhancing the security of our service infrastructure to protect Evernote and your content. There are also several important steps that you can take to ensure that your data on any site, including Evernote, is secure: First, avoid using simple passwords based on dictionary words. Never use the same password on multiple sites or services. Never click on 'reset password' requests in emails. Instead, go directly to the service.

"Thank you for taking the time to read this. We apologize for the annoyance of having to change your password; but, ultimately, we believe this simple step will result in a more secure Evernote experience. If you have any questions, please do not hesitate to contact Evernote Support. Signed, the Evernote team."

So that's textbook. That's what you do. You store your stuff, your passwords, in a good fashion. The benefit of it being salted is that we don't know whether anyone knows what the salting value is. We don't know that that escaped. And it would not be easy to determine what that is. So if someone did not know what the salt was - we also don't know if it's a per-account salt, and the salt is stored with the account information. I mean, there's a lot we don't know.

But the idea is that, for a bad guy to make use of salted leaked password information, they would need to run MD5 hashing as fast as they could, putting in dictionary words with the salt if they know what it is, see what comes out, and then look for that value in the database that they have acquired and see if they can see a match. If they do, then they know that account's password in order to log into the account. Well, so we don't know whether they know what the salt is. If it's not known, then they really don't have anything to go on. I mean, they're really stopped. If they do know what it is, and it's a

per-account salt, then even the approach of looking it up in a big dictionary isn't going to be successful because they're not going to be able to run passwords through the hash and then see if anyone has it. They would be limited to using the salt with a single account.

My point is that immediately retiring all of the passwords, which is what Evernote did, renders that escaped, that lost data useless. Now, if emails have escaped, as I think they did, then that's a problem, if people consider their email address secret. But at least it means that, instantly, every existing password was no longer valid, and users then in a distributed fashion took on the responsibility of changing their password. And as you said, Leo, apps operate by authenticating once and then receiving an authentication token which - and we've seen this in other services, where other services have asked people to change their passwords, but it was, like, sort of a little strange that all of your applications into that service continue to function after that system-wide reset. And what Evernote is doing is they're saying, okay, we're going to flush all the apps, also, and require people to update them.

Leo: So this is similar, and the response is similar, to what happened to LastPass; right? Twitter did the same thing, reset passwords.

Steve: Although Twitter is an example of not having...

Leo: Hashed.

Steve: Well, not having your applications having to reauthenticate.

Leo: Right, right, they didn't push those out.

Steve: Right.

Leo: So Evernote really did everything they could do except, unfortunately, prevent the initial hack.

Steve: Yes. And with any luck they, I mean, I guess I'm of a mind, yes, it's bad. Yes, it's hard to do this right. The larger your organization is, the bigger your systems and your network is, the more people have access, the more ways there are in. I mean, it's just it is so difficult to do this kind of security correctly. So I'm not of the mind that, if someone makes a mistake, you drop them and go somewhere else because I believe they can learn from these mistakes. I mean, they're probably more secure now than they were before, so that's more reason to stay with the people that have learned this can actually happen. And the fact that they were already using salted hashing says, okay, they understand the fundamentals. Yes, it's not SHA-1; but MD5 is fine, if it's salted and it's done right. And the evidence is it was all done right.

Leo: Now, here's a question for you. I'm thinking I store, right now, I store

passports, driver's license, credit card numbers, Social Security numbers, everything. My life is in Evernote. Should I move this over to LastPass? Would it be more secure? It would be, wouldn't it, because Evernote stores this on their servers.

Steve: Well, LastPass does, too. LastPass backs up all of the browser, the local browser blob, in...

Leo: It's all automatically encrypted.

Steve: Yeah, now, see, we don't know, I know nothing about Evernote's...

Leo: It's not encrypted. Well, it's cleartext on my computer unless I explicitly encrypt it. And that you have to do note by note. And furthermore, I don't know how they encrypt. So I'm thinking LastPass might be a better choice.

Steve: I trust them. We know the technology for LastPass because Joe has laid it out for me and us, and it's been vetted, and, I mean, it's bulletproof. I haven't looked to see whether Evernote has documented their technology. If they haven't, I would not consider using them. If they have, I just haven't looked.

Leo: I'm think I'm going to move all of that stuff that would really be horrendous if it were stolen. I think I'll move that.

Steve: You know, Leo, they must be doing something good.

Leo: Evernote uses 64-bit, according to Fred Flintstone in our chatroom, he's linking to an Evernote page, 64-bit RC2. So that's secure.

Steve: 64-bit? That's weird.

Leo: That's a little low, isn't it.

Steve: Yeah, it is a little low.

Leo: Should be bigger.

Steve: And RC2, as long as they do it right, it's secure. But the big advantage is...

Leo: I'm going to move it over to LastPass [laughing]. Is that right, 64-bit? That

doesn't sound right. It should be much higher; right?

Steve: Yeah, that's not many bits. So...

Leo: What type of encryption does Evernote do? Yeah, 64-bit. "We derive a 64-bit RC2 key from your passphrase and use this to encrypt the text. This is the longest symmetric key length permitted by U.S. Export."

Steve: Oh, my goodness. That's why.

Leo: That's why.

Steve: That's no longer true, though, Leo.

Leo: I thought that that changed.

Steve: Yeah. Used to be 40-bit. In the first place, it was 40, not 64. And that hasn't been true for years.

Leo: "We do not receive any copy of the key or your passphrase or any escrow mechanism to recover your encrypted data. If you forget your passphrase, we can't recover it."

Steve: That's all good.

Leo: "User authentication is performed over SSL. Data in user notes is also transferred via SSL." But unless you explicitly encrypt it, it is not, it's in the clear after it gets to the servers.

Steve: Wow. Wow.

Leo: "Several of the company's founders come from a strong encryption background, founders of CoreStreet. For Evernote's consumer product, the current encryption algorithms are chosen more for exportability under the Commerce Department rather than strength, since our software permits the encryption of arbitrary user data with no escrow. We'd be interested in offering something stronger when we have the staffing to fight the lengthy export battle."

Steve: Wow. I wonder if that's - I thought there were no export restrictions on crypto now. I mean, everybody's doing 256-bit AES, which blows away 64-bit RC2.

Leo: I would think.

Steve: Yeah. I mean, you've even got AES supported in all the chips now. So it's no longer in - there's no way that it's slow because you've got hardware-level support for it.

Leo: And everything I put in a LastPass secure note is by default encrypted, and I've got two-factor authentication turned on, which Evernote will offer but doesn't currently offer.

Steve: Correct. In fact, I was going to mention that they're - they've announced this week, as a consequence of last week's breach, that they are going to accelerate their plans to offer two-factor authentication. We don't know anything more about what it would be, maybe phone text message loop or who knows. Maybe hardware token, but that's a problem on smartphones.

Leo: I'm inclined to move everything over to LastPass. I just feel better about it.

Steve: I'm quite happy with LastPass.

Leo: So do you store online things like that?

Steve: Yeah.

Leo: When you travel places, you're supposed to take with you a copy of your passport. Rather than do that, I take a PDF on my phone, encrypted PDF on my phone in Evernote. But I think I'm going to move it over to LastPass.

Steve: Yeah, I tell you, I'm actually still using Jungle Disk and really happy with it. I've been very impressed. I kind of went back and took a look at it, and it moves everything up to Amazon. I mean, I know the architecture because Jungle Dave told us all about it back in the day. And in fact Rackspace purchased Jungle Disk from Jungle Dave.

Leo: All right. Move on. I just wanted some advice on that. I think I am going to move everything, all that stuff into LastPass.

Steve: It's hard to see how LastPass could have a problem that, due to the architecture, they absolutely don't want the responsibility, and it's not possible to store anything in LastPass that isn't really strongly encrypted.

Leo: Right. I like that.

Steve: The other little bit of news I thought was odd, and this isn't like a problem in the wild. But it could be soon. A developer noted that only one browser in the industry had abided by published recommendations for limiting the amount of the local storage, which is a new feature of HTML5. So traditionally the way a remote site could store data was with a cookie. Not very convenient because it has to go back and forth with request headers and reply headers. But cookie storage was limited to about 4K, sort of by agreed spec, which is plenty of data if you just want to store a gibberishy-looking token to represent who you are, that you're logged on, you're authenticated and so forth. And you can store useful information in addition, and 4K is plenty of space.

But the developers of the web said, you know, as we move more towards an application-centric, browser-based platform, it might be useful and, lord, I mean, whose systems don't have gigabytes of data now, it might be useful to allow something called "local storage." So there's a spec, the standard W3C specification under HTML5, for something called "web storage." And this allows between 2.5 and 10MB per domain name. For example, Chrome went low. They set their limit at two. Firefox set theirs at 5. And IE sets theirs at 10.

But in the specification at W3.org under the web storage facility, under their category of disk space, they say "User agents" - meaning browsers - "should limit the total amount of space allowed for storage areas. User agents should guard against sites storing data under the origin's other affiliated sites, for example, storing up to the limit in a1.example.com, a2.example.com, a3.example.com, et cetera, thus circumventing the main example.com storage limit. User agents may prompt the user when quotas are reached, allowing the user to grant a site more space. This enables sites to store many user-created documents on the user's computer, for instance. User agents should allow users to see how much space each domain is using. A mostly arbitrary limit of 5MB per origin is recommended. Implementation feedback is welcomed and will be used to update this suggestion in the future."

Now, only Firefox heeded that warning. And a developer created a proof-of-concept which does exactly what the W3C org said could be done. And under Chrome 25, Safari 6, Opera 12, and IE 10, it is possible to fill up the user's hard drive at the rate of 1GB of data stored every 16 seconds. And this was done on an SSD-based MacBook Pro. So, to be more clear, if this isn't fixed soon, you could go to a malicious website that, for no other reason than it wanted to, it could completely fill your hard drive with garbage, essentially, at the rate of a gigabyte of garbage every 16 seconds.

Leo: That's not good.

Steve: That's not good.

Leo: Would it fill occupied space or just slack space?

Steve: Oh, just slack space.

Leo: Oh, okay. It's not erasing anything.

Steve: Yeah. So it's, yeah, it's not erasing anything. But most people, hopefully, if

they've got their terabyte drives, they've got room, and this thing would just take it all up. So anyway, this was - I got a lot of tweets about this. That's the story about that. I'm sure - oh, I should mention also that there is a - there's a little bit of a gotcha because in some cases subdomains are valid. For example, GitHub uses subdomains of GitHub.com to store legitimate individual user data. And Appshot does the same thing, and many others. So the problem is that it's not clear how this works. If you absolutely limited storage for the second-level domain, like "example" of example.com, then all of its subdomains would have to share that storage, when in fact it might not be abusive. It might be legitimate. Yet there's no mechanism, I mean, we really have a problem here because, unless the whole browser had a limit, but then you'd have a limit on how many sites could do this.

So unfortunately there's a bit of a conundrum here that we haven't come up with a solution for, other than - and, see, they suggest involving the user. Oh, my god. That's not going to work. That's never going to turn out well, where dialogue boxes are popping up saying, you know, this site would like to exceed its limits on the amount of storage it can use on your computer. People are going to go, what? I mean, you'd think that was malware. I mean, our listeners would, anyway, and they would be right to be suspicious of that.

So it's not easy. It's not clear what kind of a solution we have. I mean, unfortunately, I think the idea of having websites able to arbitrarily store multiple megabytes of data on people's computers is just probably fundamentally bad. I'm not sure that I think that's a good idea at all. So maybe that's why we're in trouble is this was never a good idea.

And there was a little bit of interesting news on the Do Not Track header front, which I thought was interesting. This is proceeding exactly as I predicted it would. SANS reported a story that I think they picked up from Computerworld, if I remember, saying that groups representing the interests of Internet companies are speaking out against proposed legislation - meaning, okay, legislation is getting proposed, which was inevitable - that would require all online companies to honor Do Not Track requests from consumers. One of the bill's sponsors, Senator Jay Rockefeller, said that companies are not currently honoring those requests; whereas Lou Mastria, managing director of the Digital Advertising Alliance, which you can imagine is on the other side of this argument, disagrees, saying that the bill is unnecessary because self-regulation is working.

Leo: That's always good.

Steve: Uh-huh.

Leo: Hey, we're self-regulating, you know?

Steve: Don't make us do it. We'll do it ourselves. Ah, yeah. Good luck with that. Technology-oriented think-tank The Information Technology and Innovation Foundation, which of course tells you nothing about what interests they represent, noted that Do Not Track legislation could be ultimately detrimental to consumers - okay, now we know whose side they're on - because a significant amount of web content is supported by targeted Internet advertising. And see, that's, to me, that's the big question, whether that's really true or not. It's not clear to me that profiling ever did work. Yes, people like to have it, but it's not clear that they need it.

The proposed legislation would allow the U.S. Federal Trade Commission, our FTC, to enforce action against companies that do not comply with consumer Do Not Track requests and would restrict online companies to collect only the data necessary to deliver their content or services. So, and then that puts the Apache web server in an interesting position, doesn't it, because it's...

Leo: Now, what is the current status of that? Because I got a number of, well, one, tweet from a guy who said, "You guys are idiots. You don't understand at all."

Steve: Yeah, I know. I saw that tweet, Leo. He sent it to both of us.

Leo: I think he doesn't understand. But...

Steve: He's a UNIX weenie, and I understand that such people have strong feelings that aren't in the real world.

Leo: Well, he was defend- I guess he wasn't disagreeing with us. He was implying that we kind of misunderstood the intent, which we didn't.

Steve: No, well, he was arguing that, because IE 10 breaks the rules, Apache has the right, the server has the right to prevent that from being seen by the application on the server. And that is ridiculous. The application could certainly make that decision. The server, I mean, it's pure political pique on some Apache developer's part, to preemptively remove information that the user's browser is sending to the application behind the server. It's ridiculous.

Leo: That makes sense.

Steve: Yeah. Sorry about that. I just feel strongly.

Leo: Actually.

Steve: Let's see how this turns out.

Leo: Yeah, I mean, you know me. I kind of defend the need for...

Steve: The need.

Leo: The need because this is sites that are giving you free services, whether you understand it or not, are making the...

Steve: They will be just fine, Leo.

Leo: And that's, I think, true because I think tracking doesn't in fact make the ads any better. So...

Steve: Correct. Correct.

Leo: But I think that - I don't know if I want to see governmental regulations against companies. You know, the next, I mean, maybe they'll say, oh, you shouldn't have ads at all.

Steve: No. They're not going to say that. It's like, you know, they're not going to take away your guns if they say that we'd rather you didn't have submachine guns in your hip pocket.

Leo: I guess the presumption is that tracking is something that's done without the user's knowledge. And I think that that's probably an appropriate thing to protect.

Steve: People who used to be in the advertising industry have said, if we knew what was really being done, we'd all be screaming with our hair on fire. So in fact there was some guy...

Leo: I don't know if that's true, but...

Steve: I know. We just need to have him talk to you, Leo. I'll find him.

Leo: I don't think that's true. That's a popular thing to say, but I think there's - I think just as there are - there's paranoia on both sides, let's put it that way. But, you know, there's a lot of wonderful free web services that are ad supported, like this one, like this show. Now, we don't use tracking cookies. We don't need to. But there are wonderful, you know, Facebook and Google and so forth, that people are getting for free, and it certainly should be their understanding that there's got to be some sort of payment for that.

Steve: Leo, I have no problem with ads. I just have a problem with ads that surreptitiously link people across websites. That's the problem. And because it is being done without their knowledge and consent. And I think if we broke that linkage, nothing would happen.

Leo: Well, I think that that's true. But I don't think that that's enough reason to say let's turn it off. I mean, I agree with you that tracking doesn't seem to really have any value at all. The other thing...

Steve: But that's all we're turning off. We're just turning off tracking.

Leo: But that's not for the government to decide whether it has value to the company that wants to use it. I think what you probably more properly could say is you've got to explain what's happening on a page to the viewer or the user so that they understand.

Steve: Well, about if - how about the government saying, if a user asserts they do not wish to be tracked...

Leo: Then don't track them. I think that's fine.

Steve: Okay. And that's really all we're saying.

Leo: I think that's fine. I think that's fine. And by the way, as a result of all of this paranoia, justified or not, we're putting a cookie policy on our page because we do use tracking cookies in some ways. But, I mean, we use them, I think, in a benign way, for instance Google Analytics. But people are very paranoid about it all. So...

Steve: Well, and what we need to see is you need to go to a site that says: Hi there. We notice that your browser has a Do Not Track header asking that no tracking be done. This site is...

Leo: But, see, this is a little unfair. So let's say I say I don't want any tracking. And I don't think it's for the government to decide whether Facebook needs tracking or not. But let's say I don't want - the government says I should be able to say I don't want to be tracked, and I want to use Facebook for free.

Steve: Leo, Leo, Leo, it's not Facebook that's tracking. It's their advertisers. Facebook has nothing to do with this. Facebook can't stop it. Facebook is out of the...

Leo: But understand they charge a certain amount to their advertisers, and the presumption is that the advertisers are getting their value. Let's say - I don't think it's for the government or you or me to say whether that value implies tracking or not. I think what this really is, is like Adblock. It's like enforcing the right to ad-block. It's saying you should be able to use Facebook for free, whether they like it or not. And I don't think that's right. So I guess what will start happening, and I think this almost certainly will be what happens, is that you will be offered the right not to - to not have tracking. But they will also say, but then you can't use our service.

Steve: Yeah. And that's fine. And that was what I was about to say, or that's what I was saying, was that you'll go to a site, and it'll say: Hi there. You have Do Not Track on, yet this site depends upon the additional revenue that's generated by tracking you in order to function. So go into your browser and turn that off right now, and then we'll happily allow you in. And then...

Leo: And that's what you're going to see, which is a fragmenting of the web. But that's what you're going to see.

Steve: Yeah. And there will be people who'll go, oh, I'd rather go somewhere else. And there will be people who go, oh, I didn't realize. I'll turn that off for you, no problem. Yeah, I mean, I think that's fine.

Leo: I think it's probably all right.

Steve: So I had a tweet from someone asking me something I didn't know, which is what's the URL for signing up for Audible and giving credit to Security Now!.

Leo: AudiblePodcast.com/securitynow.

Steve: Securitynow, just all one word.

Leo: Yeah, but they'll be tracking you as a result.

Steve: [Laughing]

Leo: That's called "tracking you"; right? But that's it. AudiblePodcast - in fact, while we're doing it I'll do the Audible ad. AudiblePodcast.com/securitynow. It's a very benign sort of tracking. It's kind of an opt-in tracking because obviously, if you go to that URL, you're sending a signal intentionally to Audible, hey, I saw this on Security Now!. But I guess that's a little different.

Steve: It's completely different. That's not tracking at all. Tracking is linking people across websites. That's what everyone objects to. And aggregating a database about people's identities, anonymous or not, and often not, in order to build a profile of them. And people are saying, eh, we'd rather not have that.

Leo: Right. Understandable.

Steve: Yeah.

Leo: And I don't think that asking people to go to our URL is in any way an invasion of their privacy. In fact, what we're saying is let them know you heard about it on Security Now!.

Steve: Yeah.

Leo: Yeah. I think that's fine.

Steve: I heard from, actually just this morning, Brian Hall, who is a frequent tweeter to me. He's "bhall7x." He just sent back, he said, "WizMouse is awesome. Been using it for about a year on Windows 7 and love it." And our listeners will remember that KatMouse is what I had previously recommended, but another listener in our Q&A, I think last week, said, hey, WizMouse - oh, yeah, it was our bonus question No. 11 - said WizMouse solves the problem of being able to use your mouse wheel over whatever window you're currently hovering over without needing to click and activate it. And it takes away the third button, the middle mouse wheel button functionality, which he always was turning off. So if you use that, stay with KatMouse. If you don't, apparently WizMouse is better. I still haven't gotten around to using it or to looking at it. But I just wanted to pass that on. And also, despite your pooh-poohing my music choice, Leo...

Leo: [Grunting]

Steve: Many tweets have come back who checked out Chuck Wild's Liquid Mind series of 10 CDs. And one good friend, who's a coder, commented that it's the only music he has ever been able to code to. So, yes, it does - I know you think of it as putting you into a coma rather than into a state of relaxation, but I didn't want listeners to be discouraged by your opinion, since others have liked it.

Leo: Oh, I didn't say it was that - did I really say it was that bad?

Steve: Oh, yeah, yeah. Yeah, you did. You started to snore.

Leo: [Laughing]

Steve: And I got a nice note dated, yeah, March 4th, just Monday, from a Chartier, I'm sorry, Gabriel Chartier, because he reversed his last name and first name in the From line, titled "SpinRite Testimonial." He said, "Hi, Steve. I'm a long-time Security Now! fanboy and SpinRite user. In Episode 386 you mentioned a testimonial about a user accessing raw data with SpinRite. I have had excellent results with a USB SATA IDE dock that I have mapped over a USB line to a DOS VM in VirtualBox. I lock a drive and start the virtual machine, and I am off and rolling, all along giving me full access to my host system while SpinRite chugs away in the background on the external drive.

"Testimonial," he says. "I saved a friend's data. Larry, the owner of a golf shop, had all of his tax info on his laptop." Yeah, it's that time of year, folks. "After letting his laptop sit for a little while," he says, "(six months), when they tried to boot up, Windows gave them the unmountable boot volume error." Yeah, we hate those. "I knew right away what to do. I ran SpinRite for about 30 minutes, and it completed the Level 4 scan on the hard disk drive. Error repaired. I booted the system, ran a disk defrag analysis using the portable JkDefrag from PortableApps.com. 95 percent fragmented. So I ran a defrag and gave the system back. They are now able to do their taxes. They are very thankful to me and you, the creator of this great tool. Regards, Gabriel Chartier in Loveland, Colorado.

Leo: That's really nice.

Steve: So Gabriel, thanks for sharing that.

Leo: Excellent.

Steve: So, are you ready?

Leo: I'm ready for the Tor.

Steve: [Chuckling] Okay.

Leo: Tor 2.0.

Steve: Yeah. So everything changed since we talked about it last. The way it used to work, real quick reminder, is it used to be purely a user-anonymizing service, allowing people who wanted to access Internet servers or services out on the public Internet to do so without their IP and thus location, ISP, and potentially their identity being known. The way that was done was by hopping among a network of Tor nodes. Each node is sort of a Tor server, and they are all interconnected. And the idea is that we've all seen the sci-fi movies where - actually not even science fiction anymore - where bad guys bounce around through multiple servers out on the Internet and then finally do whatever they're going to do. Often on TV they're making a phone call bouncing off of some satellites and things, which makes it difficult to figure out that they're actually next door.

So what was so cool about this, the word "TOR" was an acronym originally, stands for The Onion Router. And it actually was a follow-on to an onion router network, which was the first thing created. And, oddly, there's sort of some strange U.S. government entanglement here. I mean, even now it's under the auspices of the Naval Research Lab. And it was originally created out of ARPA and DARPA.

Leo: Oh, now I'm suspicious.

Steve: Well, no, it's because - I think it's because we, too, need anonymity. I mean, like, wouldn't the NSA like to be able to do things and not have it traceable back to them? And so what we have is a - I mean, and Leo, it's all open source. I mean, everyone knows exactly what these Tor nodes are running. It's all done academically. So there's no way to be suspicious of this. It's that there are - it's not only people, individual users who want anonymity. It's, you know, the EFF has been a supporter of this, and the Free Software Foundation gave them an award a couple years ago for advancing the free social interests of technology on the Internet, to allow people in repressive governments to be able to have free speech. So there's a lot of good here. But to me it's interesting that it was, and still is, there's some governmental support behind this thing being developed, which I think is interesting.

Leo: General Petraeus should have used it.

Steve: The way it originally worked is a client would pick at random a few Tor nodes, and each Tor node advertised or published its public key. So the client - there is an onion router node, but then there's also sort of a - there's software that you run in your machine which is your interface to the Tor network. So a couple of nodes out of the cloud of available Tor onion routers would be chosen, and their public keys would be gathered.

The user then takes the payload that they want to put out anonymously on the Internet. And by that I mean a request for a page on a website somewhere, or a posting to a socially controversially site, or who knows what, whatever it is they want to hide the origin as being themselves. They encrypt that with the last node that they're going to visit, the so-called "exit node" where their data will finally leave the Internet. Then they put on the out - so they encrypt that payload. And sort of think of it as like sticking it into an envelope, or it's called an "onion router" because of layers of an onion. So you could think of it like maybe being a sphere, and you wrap it with encryption so that no one can get in.

Then on the outside of that you put the address of that node, that last node. Then you go to the node that will precede that one, the node that will be previous to that as you're sending stuff out. And again, you know what its public key is, so you use its public key to encrypt that sphere. And then on the outside of that you put its address. And then you choose the third one, maybe it's the first one you're going to send something to, and you encrypt what you have so far with its public key.

Now you've got this multilayered blob, and you send it to the first node. So first of all, that connection out to some random Tor node somewhere is encrypted under its public key, and only it has the matching private key. So no force on Earth that we're aware of is able to access that. But even if someone did, it's still got, remember, two more layers of encryption in these nested layers of onion in order to actually get down to the payload.

[Barely audible "yabba-dabba do"]

Leo: You're selling a lot of SpinRite today.

Steve: [Laughing] So what actually happens is that first node, since the outer shell was encrypted with its public key, the first node to receive it is able to use its private key to essentially unwrap that layer, to decrypt that layer. Then what it finds is, remember, on the outside is the address of the next node. It can't decrypt the contents inside because only the next node has that. But on the outside is the address. So it sends this blob, which is still opaque to it, on to the second node. The second node receives it, and now the outer layer was encrypted with its public key. So it uses its matching private key to decrypt that layer, which exposes the address of the third node. And underneath that is the remaining encryption, under that third node's public key. So it has no access to that, either.

And notice that that second node, the one in the middle? It only knows that it got a blob from another onion router, but it has no knowledge of where that came from. That is, it doesn't know who sent it to that onion router. So we're going to see the number three a lot because that sort of seems to be, like, the sufficient number of nodes involved to create anonymity no matter which way you come at it and how you look at it because

that second guy receives it from the first one, doesn't know who the first one received it from, and all it knows is to send it on to the third one. And then the third one, similarly, only knows that it received it from the second one, has no idea whether it came from an individual or from another onion router because all trace of the earlier hops have been removed.

Those previous shells, if we look at it like a spherical onion, those shells have been decrypted and stripped off and removed, removing all trace of its history. And at no point along the way has anyone been able to see what this thing - have any of the intermediary nodes, onion router nodes, been able to see what this thing contains. Finally, it gets to the third hop, which was, remember, the first encryption done of the actual payload. So that final onion router decrypts it, as only it's able to because only it has its private key that matches the public key that it advertises. So it decrypts it. And now here's a standard Internet packet which it simply drops out on the Internet. That is the so-called exit node from the Tor network, and the packet goes off, as packets do, bouncing around regular routers toward its destination. So that's the Tor system as it was.

Now, there was a redesign to address some theoretical problems with this. There were various people who said, well, you know, even if you couldn't encrypt the data, if you stored the data, then if at any future time it was ever possible to determine the private key from the past of an onion router, and if you happened to get the private keys associated with the paths that that blob took, then you could decrypt it and obtain all the information. So what the system used to lack is something known as "perfect forward secrecy." And we've talked about what perfect forward secrecy is. It is this property that you should not be able to learn something in the future which allows you to go back into the past and obtain knowledge that the past wouldn't want you to have. So the system was reengineered, and many other capabilities, including what we're about to discuss here, hidden services, occurred. So here's how this was reengineered.

Instead of the user in advance choosing the route that the payload is going to take and building up an onion and then just dropping it onto the first onion router and off it goes, the capabilities in the onion routers were increased. Notice that, with what I just described, the original system, there was no state being maintained. There was no notion of connections or persistent state. That is, the onion router would just sort of sit there. It didn't have to have any history at all. And if a blob came in, it would see whether it was able to decrypt it using its private key; and, if so, it would then send the blob on to the next jump in the router, or out onto the Internet, if it happened to be the last one to decrypt the innermost shell. No knowledge was necessary.

This changed with the rewrite of Tor this move to 2.0. Now what happens is that the user establishes a session with the first onion router node. It sends it - all the onion routers still broadcast a public key. But unlike the original system, where that public key was used for encryption, and that was the weakness, instead we now use session keys that are established with the so-called Diffie-Hellman handshake. And we're talked about the way that works. That's the exponentiation approach where anyone can observe the traffic going in both directions, but even the person observing it is unable to determine what the mutually agreed upon key is from looking at all of the dialogue back and forth. And this, for example, is an option now in SSL and TLS protocols and very well established, very secure, very quick and lightweight, too.

So the client sends a - it gets the first onion router's public key, and it sends the first half of this Diffie-Hellman handshake encrypted under that first router's public key. So the first router receives that. It decrypts it, and only it's able to, using its private key. And then it has the first half of this two-way handshake. It finishes the handshake, sends

back the second half of the handshake to the client, and hashes the resulting session key and signs it with it - okay, I'm sorry, I went a little fast. So when it finishes the handshake, now it has the session key.

Once the client receives that second half of the handshake, the client also will have the matching session key for the encryption. So to prove that the onion router also has it, the onion router hashes that key and then signs it with its private key, which only it has, and sends that packet back. So the client now receives that, which it believes came from the onion router that it's trying to establish a connection to. It decrypts that using that router's public key, which will only work if it was signed with the router's private key. That returns - that finishes the handshake, allows it to establish the secret pseudorandom session key that they will be using to communicate henceforth, and it's able to verify that they both had the same session key by decrypting the hash of the key.

So now we've got - we've securely and with authentication of the onion router established a connection, sort of the first link. Now the client chooses another onion router on the network, but does not contact it directly. Instead it uses the link that it's established to the first onion router to do exactly the same thing that it just did with the first one, essentially sort of virtually moving itself to the other side of that first link. So now the second link is similarly established between the first onion router and the second onion router, using exactly the same protocol. And then it moves itself out to the second onion router to establish a third link, and so on. Typically, as I said before, three links.

So that's the approach used. It's still the case that the data is bouncing from one onion router to the next. And as a consequence of moving up this chain of links, a series of session keys has been created for encrypting each link. And that same session key is used for creating an onion-wrapped packet, which is then sent down the link. And as it bounces, each session key is used to decrypt the successive layers. And since the session keys are ephemeral, as the term is, they're only used briefly for that connection, at no point in the future would compromise of a router's private key give anybody any information about how to decrypt data that was stored, moving through the onion router network.

So that's the difference. So it's subtle, but it's significant. And essentially what's been done is a more sophisticated protocol has been created. So what we have now is essentially what we had before, that is, the ability for a user to hide themselves on the network. What we want is something new, which is the ability of services like websites, IRC chat servers, meeting areas, whatever, Internet services that are traditionally always public because the only way to reach them is over the public Internet with an IP address. The gurus of Tor said, you know, we could hide these. We could use the Tor system to allow people to publish public services, meaning available to anyone who uses Tor, so that the services were available from within the Tor network and not publicly. And the only way to get to them is through the Tor network, meaning that they do not have public IP addresses and that no one using them can gain any knowledge of where they are, where they physically exist in the world, where they physically exist on the Internet. So it's sort of the next-generation evolution of Tor.

The way it works is this. Somebody who wants to offer a service of any sort, any kind of Internet service that anonymous users can access and that itself wants to remain anonymous, chooses three onion router nodes at random, just three. And these are called "introduction points." So we have introduction points are, for a given service, are three routers, three onion routers on the network. The service creates a long-term public key pair so that it can have a public key which is used within the network to identify its service. So this public key represents this service. The service establishes circuits, static circuits that continue to exist with each of these three introduction points. And we're

using three for redundancy, so that if one was off the 'Net for a while, or disappeared, it would always be possible to pick a third one. But in the meantime, you'd like to have a way to get to the service. So three gives us redundancy.

So it establishes a circuit to these introduction points using the new protocol I just explained, successive links that protect its identity. Notice that its identity is being protected now as a service in the same way that the user's identity was being protected as a client. So it establishes, over three hops, links to introduction points. And it says, here is my public key for anybody who wants to access me. Let's keep this circuit open. And if anyone wants to access me, they will reference this public key.

Then the service, in a Tor directory, publishes its public key and the identity of the three introduction points it has chosen. It signs that and encrypts it with its public key to give it authentication. And a 16-character token is statically created from that information. And that's what's known to the public. So the address of the service is this random 16-character token dot onion. That's like the domain name with the Tor system, 16 characters that are derived from its directory entry dot onion, in the same way that we have dot org and dot net and dot com.

So now, out of band, not using the Tor system at all, some user has been notified or heard that there's a service hiding within the Tor network which offers something that the user wants. So the client now, it uses its Tor interface to say, I want to connect to this service. And so it uses that funky 16-character token dot onion in order to initiate the connection. What it does is it goes to the Tor directory and uses that to look up the information it needs. It gets a one-time cookie, a so-called "nonce," n-o-n-c-e, nonce, which will represent itself just for the purpose of authentication.

Now it chooses at random a node in the Tor network which we call the "rendezvous point." And so it opens a circuit, three hops, out to a node, which is the so-called rendezvous point. And so it connects to it, and it says, hey, I would like to rendezvous with this service. Here's the service's descriptor. Please set that up for me. So, I'm sorry, it establishes the rendezvous point as the place where the service is going to connect. So three hops out to the rendezvous point. Then it talks to, it connects to the introduction point, which are sort of like the three listening outposts for the service, and it says to the introduction point: Hi there. I'm waiting at the rendezvous point. Since you're representing the service, you're one of the three rendezvous points the service has chosen. Please notify the service of my interest in connecting to it, and I'll be over at the rendezvous point.

So the introduction point, one of those three introduction points that has that static circuit up and connected to the service, it sends this information to the service along with that cookie, remember, that it's been given by the client. The service then is able to decide if it wishes to accept the connection. So this provides insulation against any kind of denial of service attack or bandwidth flooding attack and so forth. Nothing is able to get to it directly. It essentially has to mutually agree with the client that it wants to offer the service.

So assuming that it does, it then establishes a multi-hop circuit to the rendezvous point, so it'll have its three links anonymizing it again to the rendezvous. It gets there, hands the rendezvous its agreement to connect, along with the cookie, the pseudorandom cookie that the client gave. The rendezvous point forwards that to the client because the client has a connection to it that was established first. The client receives back the cookie which it initially gave. And there's, you know, wrappers of encryption on all of this. And that cookie was signed then, before it was sent back, signed by the advertised service's private key so that only its matching advertised public key is able to decrypt it, thus

authenticating the fact that it has received the cookie back from the real service. Now, the client and the service both have mutually anonymizing multi-hop connections to the rendezvous router, and it passes traffic back and forth between them anonymously. And that's how the new system works.

Leo: Well, that was simple enough. I don't know why anybody would be at all confused by that.

Steve: [Laughing] So backing off from this a little bit, essentially what this means - oh, and I should say that the standard Tor client software, the software that people get from Tor to load on their computer to access the Tor network, it is the hidden service software. That is, it's the same package. So anybody who has a computer that wanted to publish information anonymously on the Internet through Tor is able to use the same - it's not like you have to go get something secret or special or something somewhere. Everybody has it. So anyone can be a publisher and can essentially establish introduction points, publish their identity on the directory, and then arrange through some out-of-band means, on a mailing list or who knows where, or put it on some other website, here's a service that we're offering at this funky address. And then people are able to connect to it. They're anonymous, and it's anonymous, and everything is encrypted from end to end. And it's done in a way that thwarts everyone's best effort to attack this.

Leo: So this sounds like a big improvement.

Steve: It's a huge improvement. Yes. You take sort of the original concept and implement it and then think about it.

Leo: Address the issues that we've mentioned many times with it.

Steve: Yes. Yes.

Leo: Yeah. Do you feel this addresses those issues, the endpoint issues, things like that?

Steve: Yeah, the only thing that they're still - the only weakness that we know of is that it still wants to be fast. That is, it wants to send...

Leo: Good luck.

Steve: Well, yes. It's certainly, you know...

Leo: Kind of the nature of this, because of the layers, that it's going to be a little slower than direct.

Steve: Well, but here's the part, is that, if it's not heavily used, then people monitoring traffic going in and out can associate, sort of statistically associate the events of traffic entering and exiting this large cloud of onion routers. Even though it's bouncing around for a while, I mean, still, if it's heavily used, then it becomes much less easy to disambiguate entry and exit events.

Now, I mean, this is well-known. So, for example, all it would have to do is introduce arbitrary delays in the forwarding of traffic from one onion router to the next. And then it would completely break the ability to use the temporal connectivity of traffic entering and exiting the cloud to associate those events. On the other hand, it would really make it slow. So they've said, no, that's not what we're going to do. The bigger the network is, the more Tor nodes there are, the more busy it becomes over time, they're recognizing that while it's in its infancy with a few nodes and not much traffic, it's easier to associate incoming and outgoing events, even though the incoming events are encrypted and the outgoing events are encrypted if they're going out to a hidden service. But still the...

Leo: They're likely to be related statistically if there's not a lot of traffic.

Steve: Yes. The relationship of them statistically is the concern. They've said, look, that's not the problem we're solving. We're going to make, again, the more nodes there are, the more popular the network is, the more use it sees, the less feasible - oh, and notice, too, in order to make the statistical process work, you really need to monitor all the nodes. Well, they're also geographically diverse. They're all over the globe now. So it makes it very difficult for any single entity to have network-level access to the entire Tor node network. I mean, many of them are deliberately in obscure locations.

So anyway, my sense is, if I were to hook up to Tor and do something through it, there's kind of a coolness factor. It's like, oh, wow, I'm really - no one can find me. It's like, well, okay, that's never a problem that I've had.

Leo: [Laughing] People can find you.

Steve: Yeah, they know where I am. But there's a coolness, there's sort of a coolness factor to it.

Leo: And Tor Mail is probably the best example of a hidden service. I think we talked about this a couple of weeks ago.

Steve: The perfect, perfect example.

Leo: And that's one where speed is not an issue.

Steve: Right.

Leo: And it's a hidden service. So, in fact, if you go to the TorMail.org page, it says,

"This ... is just an informational page. None of Tor Mail's ... systems are hosted [here] or on any server ... you can find the IP address. Seizing or shutting down this website will have no effect on Tor Mail's services."

Steve: Very cool.

Leo: Yeah. I mean, because, you know, go ahead, find us.

Steve: That's perfect.

Leo: Yeah. That's a really - I think that's a very good example of using a hidden service in Tor.

Steve: Yeah. And, you know, one of the takeaways from this is, for everyone who just completely glazed over and wished they were listening to Liquid Mind for the last 15 minutes rather than me, or their mind has been liquefied...

Leo: I have a liquid mind.

Steve: Jellified. Look at what it takes to create anonymity. That's how non-anonymous the Internet is. They didn't do this just because they were bored one afternoon. I mean, it takes that much in order to truly anonymize the Internet. So anyone who's thinking, oh, no one knows who I am, it's like, only because they don't care. But anyone who cared would absolutely know who you were. And so I think this is an interesting lesson is that this is what it takes, all of that rigmarole, in order to protect, to truly protect identities on the Internet. And even then, if the data that you send gives away who you are, well, then you're not anonymous anymore.

Leo: [Laughing] And you were worried about tracking cookies.

Steve: [Laughing]

Leo: Sorry, I had to throw that in. Steve Gibson is the guy at GRC.com, the Gibson Research Corporation. That's how you can find him on the Internet. You can also follow him on the Twitter, @SGgrc. Now, if you go there, you'll find a lot of stuff, free stuff, like his great ShieldsUP! service. But don't forget his bread and butter. The yabba-dabba dos you hear in the background all come from one and only one thing, and that's SpinRite, the world's best hard drive and maintenance utility. You've got to have it if you have a hard drive. So pick up a copy.

Steve: And for many of our really great listeners, Leo, who have just really been supporting me and my efforts and my ability to do this and offer all this stuff for free...

Leo: Oh, we're so pleased, yeah.

Steve: ...it comes all from SpinRite.

Leo: Steve does, on his own, make 16Kb versions available of the show, audio, so that's a very small file. He also does transcripts, the smallest file of all, human-typed, actual, legible transcripts of each and every show. Both of those...

Steve: Oh, yeah, Elaine researches them. Every single thing I mention, she goes and makes sure that it's spelled right and...

Leo: I'll give you an example. I almost can guarantee you, if you're looking at the transcript right now, Tor is spelled capital "T," lowercase "o-r" because she figured it out; right?

Steve: Yup.

Leo: That's how good she is. Elaine, good job. And if it's not, well, she's going out and fixing it right now with global search and replace [laughing].

Steve: No, she listens to this, too.

Leo: Well, of course. She has to. She's typing it.

Steve: That's right.

Leo: Yeah. And by the way, you can watch the show live. We do it live every Wednesday, 11:00 a.m. Pacific, 2:00 p.m. Eastern time. I almost had to punt today because my son Henry's giving his senior, was giving his senior speech right about now. I was just either going to leave or have Iyaz do the show. But it turns - he moved it to Monday, thank goodness. So I can go see his senior speech.

Steve: Oh, how cool. And so - and he's really come together, hasn't he.

Leo: Yeah, he has.

Steve: Yay.

Leo: He's waiting - he got into Arizona State. He's waiting to hear from Cal State

Chico and Colorado University, CU Boulder.

Steve: You don't want him to go to Chico.

Leo: They're both party schools.

Steve: Yeah.

Leo: I said, "Henry, is it a coincidence that you picked the two biggest party schools in the United States?" He said, "They are?" I said, "Yes, they are." But you know what, any college in the U.S. is a party school on a Saturday night, so I don't know if it makes any difference.

Steve: Yeah, true.

Leo: I mean, if you want to party, you'll find a party. But ASU's a great school. That's where Lawrence Krauss teaches. And he's interested in music, so he's applied to schools that have good music programs.

Steve: Oh, cool.

Leo: Any event, I got a little sidetracked. Watch the show, 11:00 a.m. Pacific, 2:00 p.m. Eastern time. That would be 1900 UTC. And it's great to have you live with the people in the chatroom. We all chat together as it's going on. But if you can't watch live, as I mentioned, on-demand versions available at Steve's site, and we have a higher-quality audio as well as video at TWiT.tv/sn, Security Now!, TWiT.tv/sn. And of course you can always get a copy wherever finer Internet broadcasts are aggregated and served.

[Laughter]

Leo: I guess that's it. Next week a Q&A, so do go to GRC.com/feedback if you have questions about security, privacy, or any of the things we talk about on the show.

Steve: Yeah.

Leo: Steve will answer 10 of them next week.

Steve: I'll read them, and we'll be back in a week.

Leo: Thank you, Steve.

Steve: Thanks, Leo.