SECURITY NOW!

Transcript of Episode #493

Tor: Not so Anonymous

Description: After catching up with a few important security events of the week, Steve and Leo revisit and dissect the anonymity promises of TOR in light of scores of academic papers which have questioned its guarantees.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-493.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-493-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson's here. We'll take a look at the week's tech news, including the latest flaw. He's got an update on glibc. We also talk about Tor. Turns out it ain't all that anonymous. Maybe you'd better be listening, next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 493, recorded February 3rd, 2015: Tor? Not so anonymous.

It's time for Security Now!, the show where we cover your security and privacy online with this man right here, Mr. Steven "Tiberius" Gibson, the guy in charge at GRC.com. Hi, Steve.

Steve Gibson: Hey, Leo. It's great to be with you again, as always.

Leo: Is this your first time watching a Super Bowl? Is that it?

Steve: You know, I don't think there was as much buzz about any previous Super Bowl. So I've been sort of warming up to it gradually, year by year. Greg, my illustrious tech support guy, who's been with me for I think 23 years, he is just like - okay. He tolerates employment just so that it allows him to support his need for sports.

Leo: That's a lot of us. It's not unusual.

Steve: You know, he's doing fantasy - I think he founded fantasy football, whatever that is, but I've been hearing about it, like, forever.

Leo: Oh, yeah. It's huge, yeah.

Steve: And so he's just like, I mean, sports is him. In the same way that coding, you know, "Born to Code" is on my T-shirt. Well, I don't know, "Born to Throw Something" is on Greg's. And so as a spectator - I don't think he's a participant, or at least not at, you know, we're all getting kind of older, and so it's not safe anymore. But so he's also a commercial fanatic. And so like the Super Bowl commercials, he was just like, all over them for, like, for years, before it even became a thing.

So anyway, this year everyone was talking, of course, about Deflategate and all that mess. And everyone was talking about the game. So I thought, well, I'll just have it on in the background while I'm working on SQRL. And I have to say I didn't much work done on SQRL. My hands, my fingers were poised over the keyboard, but I was looking over here at the screen thinking, wow, these guys are amazing. It was just - it was great. And then of course I heard that everyone agrees that it was a great game. It was like a fabulous game for me to have, like, lose my football virginity.

Leo: It was. A lot of them are not. A lot of times the Super Bowls are kind of a letdown after great playoffs and all of that. But, yeah, this was an unusually good game, and it was hard fought down to the bitter end. And I think anybody could have won it. I'm glad to...

Steve: And some amazing plays. The guy who was, like, bouncing the ball around and then finally ended up grabbing it at the end and keeping it off the ground. And it's just like, wow, these guys just don't give up.

Leo: Yeah, it's pretty incredible. Plus there's the great ads in between, which is always a lot of fun. So, yeah.

Steve: Oh, and that halftime spectacular.

Leo: I think you also saw the best halftime ever.

Steve: Wow.

Leo: I'd venture that. And it's interesting see the technology. And by the way, concerts now are doing that everywhere. Projection has become a high art. I first really kind of realized that when John took us to see "The Wall," Roger Waters' "The Wall." And when they build the wall, the projections on the wall are pretty spectacular, but now have gone since to other big-time concerts. And the projections they have, the capability they have is amazing. And you really saw the state of the art there on the stadium floor with Katy Perry on the halftime. They were used at the Olympics, I think.

Steve: Yeah. There was also a Beatles retrospective, has it been, what, 50 years or something? I think it might have been a 50th anniversary. Anyway, they used a lot of graphics. But they were just sweeping, huge, and dynamic, and well produced. No, I mean, we're just - we're beginning to get all this stuff figured out. So, yeah. But, oh.

Leo: And we should give credit to a guy named Stan Honey, apparently, who's the guy who invented the yellow first-down line and hockey puck trails.

Steve: Oh. First time I saw that on the grass I thought...

Leo: Isn't that great?

Steve: Because, I mean, yeah. As a programmer, and I've messed with visual stuff, I look at that in terms of pulling that off. And it's just like, oh, my god. And everyone else is like, yeah, okay, yeah, so? But it's like, no, write some code. Like, make that happen. It is hard.

Leo: Yeah, yeah. This guy apparently is quite the genius. The company's called Sportvision or Sportsvision.

Steve: They have all those little bots floating around over the stadium, I mean, you know, to get amazing camera angles and follow guys running down the field and everything. It's like...

Leo: It's pretty amazing.

Steve: It's just incredible stuff, yeah.

Leo: Sometimes we'll sit down, and we'll talk about the audio, as well, because Alex Lindsay has done some NFL audio, and it's really interesting what they do. They wire the shoulder pads and stuff. It's very interesting.

Steve: And, you know, cable providers' availability to turn up the bandwidth and turn down the compression, and it really looked like we were seeing sharper video for the sports event than - and of course that's traditionally been done because sports was one of the early drivers of HD sales.

Leo: Yeah. Well, there were, I understand, three or four 4K cameras at the Super Bowl, but they never used them for 4K because of course nobody has, you know, there's really no way to broadcast 4K.

Steve: Nowhere for it to go.

Leo: But what they were doing is using it to zoom in because they have - they can take one quarter of the screen, and that's 1080P. So they had 4X zoom, basically.

Steve: And speaking of which, at the very end of MacBreak Weekly, Alex was - was that an oversampling interpolating piece of software?

Leo: That's a good question. Hydra, we were talking about, a camera app for the iOS platform. And it's interesting because I don't know how it works. It shoots 50 shots of handheld, and he said even more if you're on a tripod. And somehow...

[Crosstalk]

Steve: Yup. Because, think about it, I mean, you're - if you oversample, that's what classic oversampling is. You take a huge number of photos of the same thing, you are going to be able to mathematically increase the effective resolution. So, you know, in order to interpolate. But, I mean, to see you zoom in from such a distance on the text on that screen. That's just amazing.

Leo: That was pretty impressive, wasn't it. Yeah, I have to play with this a little bit. That was Alex's pick on MacBreak Weekly, Hydra, H-Y-D-R-A.

Steve: Very cool. So...

Leo: Yeah.

Steve: Today is the long-awaited episode that is going to disappoint some people, unfortunately, who may have been over-relying on the guarantees of anonymity that the Tor network promises. There was a paper that caught my attention, which I ended up being a little disappointed in. But the references, it had, like, two pages of references to prior work. So I dug deep and looked at everything that's been done. And bottom line is I wouldn't trust Tor to my anonymity any longer. It was a nice experiment. We first talked about it in Episode 70 in March of 2008. And we've revisited it a few times. But the designers made some choices which arguably back then may have made sense. But they're now built into the system, and in 2015 you can't count on them any longer.

Leo: Oh. Oh, dear.

Steve: So we're going to look at unmasking Tor's anonymity promises. And unfortunately, there's not much left when we really dig down. But we're going to talk about the news of the week also. There were two main things that happened. And that is news of Regin's apparent heritage. And just as we were going to air last week I talked about a vulnerability that I erroneously associated with Openwall. It turns out that's just where the link came in from a reposting to the Openwall list. But it wasn't about Openwall at all. It's about mainstream Linux, and it's really bad. So we'll talk about that. I saw in the mailbag an interesting question about SQRL I wanted to talk about, and then we'll plow into Tor.

Leo: Steve Gibson, you've got the security news for us?

Steve: Yeah. So first item is that Der Spiegel released some additional Edward Snowden documents, probably now about three weeks ago, maybe a little more than that. And those discussed a project that the NSA was known to have had called QWERTY, Q-W-E-R-T-Y, named, of course, after the top row, the top left row of characters on our typical QWERTY keyboard.

In looking at the documents, a number of security researchers around the world, and specifically the guys at Kaspersky Lab, thought they looked familiar. And they dug into their Regin code and found absolute clear duplication, meaning that there is a strong reason to believe that Regin, which we were previously thinking, due to the targets that it had been aimed at, was probably not a Western tool, but may have, for example, been Russian in origin because the target seemed to be us, seemed to be more Western-oriented. But the evidence strongly implies that this is another one of the tools by the so-called Five Eyes team.

Five Eyes is Australia, Canada, New Zealand, the U.K., and the U.S., which are bound by a multilateral agreement called UKUSA, which is a treaty for joint cooperation in signals intelligence. And this specifically looks like it was originally Australian in origin, but is now tied into this product that is directly traceable back to the NSA. And in fact in the first page of our show notes, I took a snapshot of a screenshot showing side-by-side code of Regin on the left and QWERTY on the right. And that particular snapshot highlights one instruction, a push instruction. But if you look above and below, it's all the same code. And there's just no question that this came from the same source.

So even though attribution is notoriously difficult, and of course we were talking about this with regard to the Sony attack and where that came from, it really looks like there's some serious cyberespionage technology available to our forces, I mean, to our governments, and being shared among these countries. And remember that Regin is the one where attacks were - after it was identified, we looked at how it was being used, and it was things like going in and getting the itineraries of guests in hotels and figuring out who was traveling and which people were meeting with each other. And, I mean, true sort of cyberespionage, where you're creeping around the Internet, pulling records out from where you need it. And so this thing has infiltrated lots of networks in the past, and it looks like it's another one of the tools that the West has available to it. So really, really much more interesting than we thought.

Okay, now, last week I had mentioned Ghost. It was a vulnerability that had just been posted. And as I mentioned at the top of the show, I mistakenly believed that it was tied to Openwall because it was in the Openwall mailing list, and that's the link that I had. And I promised to look into it for today's podcast. Well, it turns out it's just a straight-up very bad Linux vulnerability.

Leo: Oh.

Steve: Which is, yeah, which has existed since 2000 and was sort of coincidentally removed in 2012. But it existed for 12 years, and it was never perceived until researchers at Qualys just were doing an internal security audit and stumbled over this thing. Even though it was removed, because it wasn't at the time seen as a security vulnerability, it still exists in many packages which are deployed that haven't bothered to update themselves to the latest, to the very latest. It's in glibc. And it is one of the more fundamental functions that exists. It's in the function - I'm looking for it in my notes.

Leo: Glibc is on every - is everywhere.

Steve: Oh, it is. Well...

Leo: That's a library that basically you need.

Steve: Well, yes, because it provides gethostbyname function. Gethostbyname is DNS lookup. Gethostbyname, just like it sounds. Get the IP of this computer by its name. So Qualys immediately alerted the major Linux distributors about the security hole. And by the time this thing became public, most had released patches for it. It's interesting in that you would tend to think that it would be difficult to exploit. It turns out that it overwrites the heap, which is - the stack is one of the dynamic allocation structures that grows down. The heap typically grows up from the bottom of memory, where the stack grows down from the top of memory, just in terms of the way memory is allocated. And at most a character pointer can be overwritten.

Now, on a 32-bit machine, that's four bytes of memory. On a 64-bit machine, that's eight bytes of memory, that is, that's the natural size of a pointer on those hardware architectures. And so you'd think, eh, you know, you really can't do much with four bytes, or maybe eight bytes. And what you can overwrite is strictly limited. You can only overwrite digits and dots and a terminating null. So what that tells us is that there was a mistake in the function such that the DNS, sort of the ASCII version of the DNS IP address could, like, just barely overflow the end of some buffer, so only those characters.

Well, it turns out that, despite those limitations, arbitrary code execution can be achieved. And as a proof of concept, they developed a full-fledged remote exploit against the Exim mail server, which bypasses all existing protections - address space layout randomization, the no-execute bit on the segments, and everything. And I think it was ZDNet that wrote, "Unlike some security announcements, these guys are not crying wolf. Qualys has developed a proof of concept in which simply sending a specially created email to a mail server enabled them to create a remote shell to the [Linux] machine."

Leo: Wow.

Steve: Yeah, it's like, holy crap.

Leo: It's convenient. That's really convenient, yeah.

Steve: Yeah. So, okay. So because it wasn't seen before as a security problem, there was no move to go back and retroactively make sure that nobody was using this 12-year span of glibc libraries. So it exists in any Linux system that was built with glibc-2.2, which was released on November 10, 2000; and it was fixed in between releases of glibc versions 2.17 and 2.18. So anyway, this is the kind of thing where you absolutely want to make sure, if you have exposure to the public Internet, specifically if you're running a server, a Linux Red Hat, Debian, CentOS, Ubuntu, I mean, just across the board. As you said, Leo, glibc is intrinsic to Linux, its core networking functionality. So this was a biggie, definitely something that you want to make sure you've gotten yourself patched for.

Leo: Yeah, wow.

Steve: I ran across a fun question about SQRL that I wanted to address because it represents sort of one of the main features of the protocol which might not be clear to people. Jay Littlefield in San Francisco is a listener. He wrote, saying, "Hello, Leo and Steve." He said, "Fan of the show here. I've been listening to Security Now! on my commute for the past several years. I'm a proud owner of SpinRite and, thanks to you two, a Harry's razor convert." He said, "I really appreciate the great shows you produce. I'm also very excited about SQRL and hope to use it on my own websites as it becomes available within the major website development packages."

And that work is underway. In fact, there are guys just waiting for me to put the final, the finishing touches on the protocol because I've been adding some features and taking away some features as I've been just pushing this, like, right up to the finish line because I sort of overdesigned some things. And when it got to actually doing it, it's like, wait a minute, we really don't need this. And I want to keep things as simple as possible so that other people implementing SQRL don't need to put stuff into the protocol that we don't end up actually using.

Anyway, so he said, "Steve, I have a SQRL question for you that I've not been able to find the answer to in any of your podcasts or on your website. You often cite the fact that SQRL creates a unique public/private key pair for each individual website accessed. Because of this, a breach of one website will not compromise your identity on any other website, unlike the common practice of reusing passwords. This is a great improvement. But what about your identity on the compromised site?

"Let's say I'm an active SQRL user for all of my web transactions. I read about a major security breach at, say, Target.com, where I'm an active SQRL user. Let's assume hypothetically that their customer database has been compromised, and I'm instructed to reset my password for my account. If I'm using a password, I can do that. But passwords are supposed to be archaic once SQRL arrives. If so, then what's the SQRL equivalent of a password reset for an individual site in the event of a breach?

"You've mentioned SQRL has the ability to change your master ID should it become compromised. But a breach of my SQRL credentials at Target.com, by definition, does not compromise my identity anywhere else on the web. Can you elaborate on how this situation should be handled by a SQRL user? I'm afraid the answer to this question is currently lost on me. Thanks again for a wonderful show, all you do for the community of followers. Regards, Jay."

Okay. So what's different about SQRL as opposed to a username and password is that SQRL is a network protocol, whereas a username and password is static data that you're requiring the Target website to hold secret. And I've often said SQRL gives a website no secrets to keep, which is part of its strength. That's the large part of where its strength derives from and the reason I got so excited about this when the idea occurred to me.

So the idea is that, if Target, in Jay's example, were compromised, and their whole user database got stolen, it doesn't matter because what they get is your public key. But what someone needs to impersonate you, that is, even if they had your public key, they're not able to impersonate you at Target.com; whereas if they had your username and password, they could.

The reason is that the SQRL protocol, the core of it is a challenge to you. The website, like Target.com, sends a nonce, a random blob of gibberish, doesn't matter what it is at all. It can be encoded with specific information to make the website's job easier, but it doesn't have to be. It just has to be unique. And then the key is you sign this random blob with your private key that never leaves your device, never needs to. And that's the whole point of the protocol. You sign the blob and return the blob with your signature. And the key-signing technology verifies with the public key that you have properly signed the private key.

So it doesn't matter if bad guys get your public key for Target.com because, if they tried to impersonate you, Target would send them a blob and say, okay, prove that you have the private key matching the public key which you are claiming is yours. And no bad guy could do that. So that's the answer, Jay. And it's part of what makes this SQRL protocol so strong is that we don't give websites any secrets to keep. And we're dynamically proving over the Internet through this protocol that, yes, we know the secret that matches the public key that we gave you. So, very cool.

And I got a fun note from Igor - okay. I didn't practice his name beforehand. Koveshnikov.

Leo: Thanks, well done.

Steve: Let's hope. Sorry, Igor.

Leo: I think it's Koveshnikov.

Steve: Koveshnikov.

Leo: Or Koveshnikov. Depends on...

Steve: In New Jersey.

Leo: ...how Russian Igor wants to be.

Steve: Well, I think he'll forgive me. He says, "Hello, Steve and Leo. About a week ago my former boss brought me his laptop that wouldn't boot. I remember I installed a Crucial M4 256GB SSD in it and was puzzled what could be wrong with the SSD. When I turned on the laptop, I could get to first mouse cursor appearance in Windows, and then the hard drive light would become solid lit, and nothing would happen.

"I connected the SSD using USB-to-SATA adapter to my computer. I could read the drive, but it took me about four days to retrieve 170GB of documents. It would freeze on some files for long minutes and then would continue to copy. It behaved exactly like a damaged mechanical hard drive, just without a clicking noise. Later, it turned out that my boss followed my suggestion, and all his documents were backed up by Carbonite. Still, I always try to retrieve most data before I start playing with drives." Actually, that's all the sentence says, so I'm not sure where he was going with that.

And he says, "I always wanted to try SpinRite on SSD, but it doesn't make sense to run maintenance on them." Well, that's not true, but we'll talk about that later. "Here was my opportunity, and I used it. Started on Level 1 just to see what SpinRite makes of the drive. It ran quickly, but showed some R's, but nothing changed when I tested a boot. Level 2, however, completely fixed the drive. When I rebooted the laptop, I got the login screen, and the laptop runs just like new. It's easy to understand what happens to damaged mechanical drives," he writes, "but really hard when it comes to SSDs. My suspicion is it's software/firmware issues. Something happens to internal table of cell assignments. But then how is SpinRite able to fix it?"

So, okay. It turns out that the push for SSD density has done the same thing to solid-state technology as it has done to mechanical technology. In other words, they are cramming, arguably, more data into a small space than they should. And they've become reliant in solid-state media, just the same as they have with physical spinning media, on error correction. So that they're, like, they're on the hairy edge, sort of operating in the gray, where they know they may not be able to get some bits back, especially as those age over time on an SSD and as the technology of the SSD wears the SSD increasingly. So they fall back to algorithmic math to fix, to figure out what the missing bits that could not be read are.

And so SpinRite actually is as useful, from all of our experience, on an SSD as it is on a hard drive. And it can recover them and make them faster. And I would argue that Level 2 is perfect maintenance. It is not writing to the drive unless it needs to. So you use SpinRite to perform preventative maintenance on an SSD, and it'll keep those U's ever from showing up. Definitely worth doing.

Leo: Thank you, Igor.

Steve: Igor.

Leo: No, Igor Koveshnikov.

Steve: Igor.

Leo: Igor Koveshnikov. Let's talk Tor with Mr. Steve "Tiberius" Gibson.

Steve: Okay. So we opened the topic back on March 28 of 2008, seven years ago, with Episode 70, that was titled "Internet Anonymity." And we came back and revisited it two years ago on the 8th of March in 2013 with our episode on "Tor Hidden Services." That was the one where we sort of followed the advance of Tor where they'd added the protocol that allowed the servers themselves to be hidden. So rather than thinking of Tor sort of as a cloud, where outside the cloud you had both clients and servers, and they were all connected to the cloud, and the cloud mushed it all around so that nobody who was looking could figure out what went in and what came out where, but essentially the servers existed on regular public IPs, the hidden services changed that, hiding the servers in .onion domain names so that you got them by some cryptic URL .onion.

So that was then. Then recently a piece of research was published. And this is only the latest among scores of papers because this has really interested academia. People have been asking, well, like doing academic attacks on the protocol. And I should mention, when I mention "attacks" throughout the rest of this podcast, I don't mean malicious DDoS sorts of attacks. I'm talking about attacking the protocol, meaning academically tearing it apart, looking for weaknesses in the design and also weaknesses in the nature of what it has to live on top of, meaning the packet-switched Internet.

And it turns out that's the Achilles heel of Tor is that the Internet was never designed for anonymity. It wasn't. Back seven years ago, I was looking through some of that transcript from back then, and I talked about how an IP address isn't a person's name, but it's easily mappable to an endpoint on the Internet because the Internet was designed, back when it was first created, only with the assurance that an existing Internet address could put a packet on the Internet, and it would eventually get to the other Internet address, where the packet contained both the source IP and a destination IP. That was all it was supposed to do.

Anonymity didn't even occur to these guys. That's many generations of evolution of application of this underlying networking structure later. People start saying, eh, but you know we'd like to also - anonymity would be handy. It would come in handy for, like, people who were trying to deal with repressive regimes, who want that feature added to their networking experience for whatever reason.

So the piece of this most recent research that caught my attention claimed, it made the claim that 81% of Tor users can be deanonymized by analyzing router information. So I looked closely at that, and I was not so impressed. What these guys did, I mean, their research was good. So I don't mean that I was not impressed by their research. But it was sort of - they did things to create that claim. Their idea was that instead of performing very careful, high-resolution timing analysis of individual packets, which is what you would normally have to do in order to attack Tor at the traffic analysis side - and we'll get to that in a second.

But rather than doing that, they were using a much more sort of soft flow mapping that Cisco builds in to many of their routers, known as NetFlow. NetFlow technology does more sort of aggregate analysis. And due to its nature, you don't have the fine-grain visibility into individual packet timing that you would otherwise get if you were monitoring the actual flow. But on the other hand, it's convenient to use NetFlow because it's built into so many routers. And essentially the router is doing a lot of that job.

So these guys asked the question, could we use NetFlow, something sort of as fuzzy instead of as focused, could we or how could we use NetFlow to deanonymize Tor traffic? Sort of as an academic exercise. What they had to do in order to pull this off is deliberately interfere with the traffic coming from a server back to the user. Now, that's a powerful technique. And we're going to come back to that in a broader context also, and look at just exactly how powerful that is. But it's powerful enough that, by delaying or dropping or blocking bursts of traffic from the server, NetFlow built into routers, as fuzzy as it is, is enough. And so that's what they were saying, where they sort of came up with this broad, 81% of Tor users. I'm less impressed with that.

But what happened was that paper was full of references to the prior work that had been done. And so I spent a lot of time digging through that and came away with the unfortunate conclusion that Tor can no longer be relied upon for anonymity. That is, if you presume that someone like the NSA, who has that kind of scope and reach, if you presume that someone like that wants to penetrate the anonymity guarantees that Tor provides, the work that's been done in attacking, and I mean that in the sense of academic attacks on this question, how good is the anonymity, the work that's been done demonstrates Tor doesn't actually provide much in the way of anonymity for that class of attacker. And so it's important to understand that it definitely obfuscates your traffic. But if someone is absolutely determined to find out who you are, they probably can, that is, if a nation-state-scale actor wants that.

There was a perfect example of that was recently reported by Ars Technica, where the FBI was pursuing some people involved with what was known as SilkRoad2. And the article says, "Despite the use of Tor, FBI investigators were able to identify IP addresses that allegedly hosted and accessed [so that is both sides] hosted and accessed SilkRoad2 servers, including the Comcast-provided IP address of someone named Brian Farrell, who prosecutors said helped manage SilkRoad2.

"In the court-filed affidavit, DHS special agent Michael Larson wrote: 'From January 2014 to July of 2014, an FBI New York Source of Information [and they said in parens (SOI), so the source of information remains unknown, but SOI is a term of art] provided reliable IP addresses for Tor and hidden services such as SilkRoad2, which included its main marketplace URL, its vendor URL, its forum URL, and its support interface.'" And I cut out all the other URLs because they're just gibberish. But, for example, the support interface is "uz434sei7arqunp6.onion." So they're all like that inside of Tor, and they're all supposed to be only available to people who are inside the network, specifically not to someone you're trying to hide from.

"The SOI [that is, that Source of Information] ultimately led to the identification of SilkRoad2 servers [which are supposed to be masked] which led to the identification of at least another 17 black markets on Tor," that is, black markets operating on Tor. "The SOI also identified approximately 78 IP addresses that accessed a vendor.onion address. A user," this affidavit explains, "cannot accidentally end up on the vendor site. The site is for vendors only, and access is only give to the site by the SilkRoad2 administrators and moderators after confirmation of a significant number of successful transactions. If a user visits the vendor URL, he or she is asked for a username and password. Without that, the vendor website cannot be viewed."

So this is real-world demonstration that there exists technology for penetrating the Tor network. Now, it doesn't have to necessarily be traffic analysis. These people may have been doing other things. There have been ways that the FBI has had of, for example, using various sorts of persistent cookies using Adobe Flash and Firefox I've seen specifically named, that is, non-Tor means of deanonymizing users, in which case it doesn't matter whether they're in Tor or not. But we definitely know from the academic research that's been done of strict pattern analysis that it is possible to penetrate the guarantee that Tor provides. And the reason is the Internet, as I said earlier, was just never designed to provide anonymity, and it really doesn't. So we should look at Tor as an experiment in how could anonymity be provided. But the fact is, it is extremely difficult to actually achieve.

Now, we could break Internet communications into two broad categories, low latency and high latency. An example of a high-latency service is email, where it's a store-and-forward system. And there, because you don't need something delivered in real-time or near real-time, you can achieve a much higher level of anonymity, especially if you do other things like padding message lengths, and obviously encrypt the contents, in order to obscure when an object leaves the anonymity network and when it enters. The problem is that that's not useful for web surfing or for other applications, SSH for example, where you're sending keystrokes through an SSH tunnel, and you'd like them to get there in relatively short time so that you're able to get the answers back in relatively short time. And of course web surfing is inherently a relatively low-latency activity.

And that's really the Achilles heel of Tor is that Tor was deliberately designed - I mean, and again, we should remember this was sort of done initially as an experiment, a project to see what could be achieved. And it grew over time, and it acquired notoriety. And it does offer some guarantees, but it is far from perfect. And so no one should assume that it is perfect. And also the designers, the original designers of Tor made some assumptions and made some compromises that are now coming back to haunt us. I found a very nice summary in one of these academic papers that summed it up this way.

They wrote, "Tor aims to protect against a peculiar threat model that is unusual within the anonymous communications community." And so they made that assertion first. And then they step back a little bit to say, "It is conventional to attempt to guarantee the anonymity of users against a global passive adversary who has the ability to observe all network links. It is also customary to assume that transiting network messages can be injected, deleted, or modified; and that the attacker [again, attacker meaning someone trying to penetrate the anonymity] controls a subset of the network nodes. This models a very powerful adversary, and systems that protect against it can be assumed to be secure in a very wide range of real-world conditions." The point this paper was making was that's not Tor.

It went on to say, "Tor, on the other hand, assumes a much weaker threat model. It protects against a weaker non-global adversary," that is, an adversary who doesn't have complete visibility into the network, which as we know a contemporary powerful adversary like a nation-state actor might. And then it says, "who can only observe a fraction of the network, modify the traffic only on this fraction, and control a fraction of Tor nodes. Furthermore," the paper says, "Tor does not attempt to protect against traffic confirmation attacks, where an adversary observes two parties that he suspects to be communicating with each other to either confirm or reject this suspicion. Instead, Tor aims to make it difficult for an adversary with a poor a priori suspicion of who is communicating with whom to gain more information."

Leo: A poor AV wizzit what? You want to translate the Latin, please?

Steve: A priori, facts known ahead of, in advance of, your intent to confirm.

Leo: Ah, okay. Just checking.

Steve: So what we discussed back eight or seven years ago was the crypto model. That was the whole onion concept. And nobody has attempted to attack the crypto of Tor because, as we said then, it is fabulous. I mean, it's fun. It's solid. And just to briefly recap, because that's not where the problem is, the user chooses a circuit through a group, sort of a cloud of Tor nodes, where they first at random choose one Tor node and negotiate keys with that Tor node. For example, the Tor node will give them its public key, and nobody knows its private key. They can then use its public key to create traffic that only it is able to decrypt.

So then you can think of them using that to jump their presence to that node, where then it becomes a proxy for them. From that position, they then choose another node in the Tor network and similarly get its public key and then generate communications that only it is able to decipher. And that's going through their first link now, which only that first link can decipher, to the second link. Then they do it a third time. They then essentially move their virtual presence out to that second node, running now through two proxies. They then choose a third node, similarly negotiate it, negotiate with it.

Okay. Now what they have is the public keys for a sequence of three nodes in the Tor network. And they generate the traffic they want to make public. And they first wrap it in the third node's crypto, using its public key. Then they wrap that in the second node's crypto, using its public key. And they finally wrap that in the first node's crypto, using its public key. Thus the concept of an onion with, like, shells, successive shells of crypto.

What's cool about this is that they send this onion to the first node. The first node can only decrypt the outer shell because that's the only thing that it has the key for. It was unable to see into any of the traffic that you used to the second node. It may have seen you get that second node's public key; but as soon as you established communications with that second node, using that second node's public key, it was then blind to it. So all it can do is take that outer wrapper off of the blob that it's received.

What it finds there is instructions about where to send it, that is, to the second node. So it forwards that blob that it cannot see into because that blob is now encrypted on the outer shell with the second node's crypto. It forwards it to the second node. Second node takes that outer wrapper off. Now it's got a blob that is becoming smaller, but it has instructions to send it to the third node. The third node receives it. It takes its crypto off. And finally it gets to the original content you, the user, wanted to send, which it places on the Internet, and out it goes.

And the beauty of this architecture, which we discussed back in Episode 70, if anyone wants a deeper dive, is that the process is bidirectional, that is, when this information comes back, the nodes are able to reverse the path. No node knows anything about the process except the next node that it forwards it to because it can only see the instructions that are now on the outside after it took its layer of crypto off. So you get through this very clever cryptographic system, strong anonymity. That's the crypto model. The problem is the traffic flow model, and that's the Achilles heel.

Now, the designers understood this, and they did what they could to weaken the applicability of the traffic flow model. For example, if everything that came in was routed deterministically to some exit node, that is, a packet goes in, and that exact packet comes out, then that would be a problem. So random-length padding is added to obscure that. Some random timing is introduced throughout this entire system. And, specifically, we presume that the whole Tor system is busy. The busier it is, and the larger it is - right now it's like 5,000 nodes. Once upon a time it was 50. So it's gotten a lot bigger. That's good for everybody who wants anonymity through the network because it just means there's a lot more going on.

So you want busy Tor nodes with lots of traffic coming in and lots of traffic going out because one of the things it does is it puts every flow of communications - I just described one flow, one circuit - through this three-node Tor network. It puts them all into a round-robin queue such that every flow has sort of its own buffer. And the first thing Tor does is chop everything up into 512-byte cells. And a cell is a term of art within the Tor network. So even though packets may be larger coming in - and Tor operates over TCP, and we know TCP is flow-based. So packets of different sizes are coming into the first Tor node and being reassembled into a continuous stream.

Tor then chops those into 512-byte cells that will definitely destroy the original packet boundaries. So basically it's repacketizing the traffic into these fixed-size 512-byte cells. The fact that they're fixed size means that they no longer reveal anything about the incoming packet sizes. And then, in a round-robin fashion, it sends them back out. So nobody looking at the outside of this Tor node can easily map up the variable-length TCP packets coming in from hopefully a large number of sources, and these fixed-size 512-byte packets, which are exiting full encrypted, wrapped in some number of onion wrappings, going off to other Tor nodes. That's all they can see.

So, I mean, you can see that the people who designed this intended for it to be daunting. The problem is that this ends up being, from a traffic flow model, a bit like metadata. We've talked about the power of metadata. Even though you can't see into a flow, you can still get information from looking at the fact of the flow. So somebody who decides, okay, inside the Tor network we have no visibility into that. We've got 12-byte cell packets, encrypted, bouncing all over the place. But ultimately it has to emerge. And by looking at the aggregate short-term packet sizes and timing, and performing a statistical analysis over what goes in and what comes out, what these academic papers have shown basically over and over and over is that it is very possible to form an opinion. So the first part of the attack is having a guess.

And in fact we've seen this before. We were just talking about this when we were talking about how the Enigma cipher was broken. Basically those bombes, they were making guesses. They were saying, we know it's not this set of rotor positions; we know it's not this set of rotor positions; nor is it this set. But it might be this set. Let's try those. And so the bombe would stop, and the guys would quickly read out a candidate set and then go try it. And then the bombe would start up again, or be restarted, looking for the next set.

So with Tor, it turns out, attacks like that also work. It's possible to rule out a huge number of possible links that are in fact - where it just doesn't work, based on what somebody knows of the Tor network. But you then end up with a bunch of candidates. And then you apply the second-level attack, which is testing the candidates. And it turns out that is a huge weakness. So if an active attacker believed that two points were communicating and had the ability to deliberately introduce deterministic changes in the traffic flow, they would see evidence of that coming out the other side.

And so the final real weakness is that, if you have an active attack, rather than a passive attack, meaning that you do something to alter the traffic, you can quickly confirm or rule out assumptions of possible pairings. And when you then operate at the scale of a nation-state, with big taps into the Internet all over the place, and when Tor is an obvious Target for bad guys using it to mask their identity, you've got means, you've got motivation, and you've got budget in order to pull this off.

So I would consider that Tor is useful as part of a defense in-depth strategy. That is, I wouldn't say don't bother using it if you really want anonymity. But I would say, first of all, don't do anything illegal. Don't do anything that you wouldn't want the federal government to know about because they fit the profile of someone able to penetrate Tor. I would say, unfortunately, the dream of a Tor user of just being able to sit back in their home or apartment somewhere with their public IP address and use Tor with confidence that they're able to do anything they want to at all, and no one can get them because they're using Tor, that's just not real.

So the only way, I mean, if you absolutely need anonymity, is to roll together old-school approaches and new-school. Go somewhere to do this as far away from home as convenient. Be anonymous there. Pay with cash. Don't go somewhere familiar. Don't know anyone. Don't make any friends. Don't talk to anyone. Don't stay long. Plan ahead. Rehearse for speed. Get it done and leave. Don't do anything there that involves your own real-world identity. Pay with cash. Change the MAC address of your machine. Maybe buy a cheap laptop just for this purpose so that it knows nothing, you have no history tied to it and so forth.

And I would say, since you have control over Tor, use more than three nodes. Don't use the default settings. Use as many as you can so that you're - oh, and use widely geographically dispersed Tor nodes. Those will be slower because all the traffic bouncing around has to go through all of those locations. So, yes, it's not going to be as quick and easy. But to get anonymity, it can't be. Do what you need to do and then pack up shop and leave. So new school and old school. But unfortunately, all of the research demonstrates today that Tor was an interesting experiment. But what we know about what the NSA is capable of doing and some evidence of what has happened shows us that we just can't rely on it for one-stop shopping of being anonymous on the Internet.

Leo: Is there a - there is no technol- oh, go ahead, sorry.

Steve: I was going to say, and remember that things like cookies, persistent cookies and Verizon supercookies and all those sorts of things, unless you're very careful about stripping them out of your traffic, they're there, too.

Leo: Yeah. By the way, Verizon says it's going to give you a true opt-out. We'll see. Haven't done it yet.

Steve: Yeah, saw that, yeah. No.

Leo: Is there any way to be anonymous?

Steve: The problem is this notion of near real-time versus high-latency non-real-time. I mean, if something like Tor existed for email, then the real Achilles heel is traffic. And I don't know how you can arrange that. If your traffic disappeared into something large, where there was no further visibility, like say you disappeared behind a university-scale NAT, where all of UCLA was 10-dot, you know, inside their network, and then you also took some old-school steps to be anonymous, it's difficult to know how they could penetrate. But maybe UCLA NAT router tracks you inside. So if law enforcement came to that IP, they'd say, okay, now we need your records to figure out who inside the network was using this at this time of day. So it probably still is possible. I think anonymity is something of an illusion on the Internet, Leo.

Leo: You could. But, I mean, let's say you did use a library, and you didn't check in, and you hid, you disguised yourself, and you...

Steve: You used a hoodie,

Leo: And you keep moving, I think is obviously the key.

Steve: Yup, yup.

Leo: You could do some stuff anonymously; yeah?

Steve: I do think, I mean, basically, though, there we're saying the Internet isn't providing our anonymity, the real world is providing our anonymity.

Leo: Right. You have to be anonymous in the real world for that to work; right.

Steve: Yes.

Leo: Because if the real world knows who you are, the Internet will somehow cough that up or can be made to cough that up.

Steve: Unfortunately, I think it really can. So what I wanted to do was to basically revisit this and dispel any belief in our listeners that use Tor and you're golden. You're really not. If somebody wants to find you, they probably can.

Leo: All right. Fascinating stuff. As they're pointing out in the chatroom, Ross Ulbricht was arrested in a library. So there you go.

Steve: He stuck around too long.

Leo: Waited a little too long to move on.

Steve: You don't want to be chatting up the librarian. You want to keep your head down, get in...

Leo: Keep moving.

Steve: Oh, and don't go back.

Leo: And never go back.

Steve: Yes.

Leo: But, you know, if you watch any of these movies, the Bourne movies or whatever, there's plenty of good blueprints there for anonymity. And keep moving is a big one. And never go back. Never retrace your routes.

Steve: Yup.

Leo: Can Tor be fixed? Or is it inherently a problem?

Steve: I think this is the problem, is that it's a layer on top of a system which was never built for anonymity. And we've talked about how difficult it is to fix things that are layers on top of other things. Like a firewall is blocking traffic that wants to get through. So instead of it somehow operating in the reverse direction, where it's blessing traffic that might be permitted, it's trying to prevent something that the system was designed to do. And the Internet was never designed, never designed for anonymity. And in the face of traffic analysis, it really doesn't provide it.

Leo: That's such a good place to stop. I think that's the coda for all of this. Steve Gibson is at GRC.com. That's where you'll find SpinRite, the world's finest hard drive maintenance and recovery utility. All the work he does on SQRL, all that other pro bono stuff he's just doing because he's interested, and he wants to give it away. It's all at GRC.com, as well as this show, 16Kb audio for the bandwidth-impaired, fabulous handwritten transcriptions of each and every episode. Elaine was very grateful for the plug you gave her last week, by the way.

Steve: Well, she's earned it, so, yes.

Leo: And you'll also find a copy of this show at our spot over here, TWiT.tv/sn for Security Now!.

Steve: The mothership.

Leo: Every episode is there. And what?

Steve: The mothership.

Leo: The mothership. I thought you said something else. But okay, thank you.

Steve: TWiT.tv.

Leo: Yeah, TWiT.tv/sn, YouTube.com/securitynow, lots of places. And of course wherever you get podcasts, including iTunes. Or use those great TWiT apps. You'll find them on every platform. We didn't write them. We're just grateful that there's some great developers out there.

Steve: They work great. I use them all the time.

Leo: Yeah, Craig Mullaney and Mark Hanson, there's some great people, Dmitry Lyalin, who just do this out of the goodness of their heart. All right. We'll be back here next Wednesday, 1:00 p.m. Pacific, 4:00 p.m. Eastern time, 2100 UTC, for yet another edition, and possibly a Q&A episode.

Steve: I think so. You did say Wednesday, and you meant Tuesday.

Leo: Tuesday, pardon me, Dudesday. So if you have a question for Steve, GRC.com/feedback.

Steve: Yes.

Leo: Or you tweet him. @SGgrc is his Twitter handle. And that's another way to reach Steve Gibson. Thank you, Steve.

Steve: Thanks, Leo. Talk to you next week.