Distributed Hash Tables

Description: After catching up with a busy week in the security space, we cover an intriguing topic in fundamental distributed Internet technology, Distributed Hash Tables, which is the somewhat awkward name for distributed database technology.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-398.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-398-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. We're going to talk about something that powers everything from Bitcoin to Tor to Amazon. It's called Distributed Hash Tables. Get your propeller hats on. This is gonna be a doozy. Security Now! is next.

Leo Laporte: It's time for Security Now! with Steve Gibson, Episode 398, recorded April 3rd, 2013: Distributed Hash Tables.

It's time for Security Now!, the show that protects you, your privacy, your loved ones, your money, your bitcoin, whatever you got online. Here he is, the king, the king of all, the Explainer in Chief, Mr. Steve Gibson of GRC.com. Hi, Steve.

Steve Gibson: Hey, Leo. Great to be with you again, as every week.

Leo: You know, I mention Bitcoin because it's been really in the news of late. They exceeded a billion dollars, a billion U.S. dollars in value. Although that says more about how inflated the bitcoin has become as opposed to the actual value of it, or the amount of bitcoin in circulation.

Steve: Well, I'd be careful of the use of the word "inflated." It's the value of the bitcoin because inflation implies a judgment, and it's a relative term.

Leo: Of course.

Steve: So we know that the value of bitcoins is going to go up over time because, I mean, they've succeeded.

Leo: Is it?

Steve: Oh, yeah. Yeah, yeah, yeah. It has to happen because...

Leo: But there's more being made all the time.

Steve: But always less more. So already the curve is...

Leo: Less than the demand, you're saying.

Steve: Already the curve is flattening. Actually, technically, the curve has always been flattening.

Leo: Oh, I see what you're saying, yeah.

Steve: The rate of production has always been decreasing. And it stops in 2040, and there will never be any more.

Leo: 2140.

Steve: Is it 2140?

Leo: 2140.

Steve: Oh, okay. Well, we've got a while, a ways to go.

Leo: Unless The New Yorker got it wrong. And the reason I brought this up, there's an excellent article - there are two very good articles. There's one in Medium.com by the financial blogger for Reuters that's kind of more about current events. And then there's a very good summary article in this week's New Yorker about Bitcoin. And more than anything, that tells me Bitcoin's real, if The New Yorker's writing about Bitcoin in a long-form article. It's very interesting. And we've talked about, in fact, this is the only show I think where you'll really hear a deep discussion of the math involved, why the crypto and the math works in Bitcoin.

Steve: Well, and some of the articles that have been written have not been as skillfully assembled, and they've talked about, oh, well, why would you trust a digital currency that just can be inflated? It's like, okay, this can't be. I mean, you can't - governments can print money. There is an absolute fixed trajectory on the rate at which bitcoins will be created. And that's set in stone. That can't change. And the only thing that makes this feasible is that, because it's digital, you can have an incredibly small fraction of a bitcoin

in a transaction. So it'd be like, if you could actually never have any more money made, yet we were continuing to create value which we wish to exchange with money, then pennies would become valuable again because they would have to be. And there you would see inflation driven by the fact that the money supply is fixed, rather than it being as it is with governmental...

Leo: You can always print more paper money. But in the year 2140, approximately, there'll be 21 million bitcoins, and that's all there will ever be.

Steve: Right.

Leo: It's kind of cool. Kind of cool.

Steve: Yeah, I think it's very neat, from a technical standpoint. And the fact that it has happened, I mean, this is now, I mean, you're right, the last week of press, one thing it's done is it's raised the price of the bitcoin above a hundred dollars now.

Leo: Well, that was what was interesting about The New Yorker article. She supposed, and there is some evidence to believe, that it was Spaniards who had lost confidence in the euro because of the Cypriot crisis and so forth, that had started buying bitcoin in great numbers and had driven the price up. And it's now, believe it or not, it's been going up a lot, $137. So your 50 bitcoins - you're a rich man, Steve Gibson. You hold on to these. But then there was the heist of half a million dollars worth of bitcoin. Now it would be worth considerably more.

Steve: Well, and this is what we cover in the podcast. We cover mistakes.

Leo: Right.

Steve: Because, when you're dealing in a world of cyber, you make mistakes in cyber technology in the same way that you make mistakes if you don't have a good lock on your bank vault.

Leo: And now we presume the mistakes are being corrected. So this kind of heist won't happen again, I guess.

Steve: Yeah, and when there were...

Leo: It's not like there aren't plenty of heists of cash, by the way.

Steve: Right, right. When there were very few exchanges of bitcoin, then the probability of it being a large event and one of very few exchanges was greater. As we get thousands of bitcoin exchanges, each exchange will be smaller. And so even if one

completely got compromised, well, it's like, okay, well, they made a mistake. They're paying the bitcoin price.

Leo: Right, the iron price. It's fascinating. But I'd leave that for you to investigate in a previous episode. If you go to GRC.com or TWiT.tv/sn, you can listen to our entire episode where Steve really does explain, in a beautiful way, the ins and outs of Bitcoin. And I think that that's probably going to be, for a long time, anyway, the canonical description of Bitcoin by a third party. It's Episode 287: Bitcoin Cryptocurrency. And then I did enjoy - there's a good Wired article. There's a good, as I mentioned, Medium.com has an article today, and The New Yorker has an article. And if you want to know more, those are kind of the laypersons' discussions.

Steve: I see no reason why currency would be exempted from the phenomenon of the Internet and everything else that, I mean, the Internet, as it's becoming more pervasive, more and more of the real world is moving there. I mean, many people do their banking online now. And so why not have a coinage? It's absolutely reasonable. And the good news is we didn't have several really bad ones first. It would have been bad if several currencies had not been well engineered, came out and, like, stumbled because they were badly put together.

In that podcast you cite, Leo, you can hear me being excited because, during the prior week in preparing for it, I taught myself what the technology was. And I came away saying, oh, my goodness, they did this right. This is so cool. And so, I mean, so here, very much in the same way that there wasn't an Internet 1 that couldn't grow, and now we have Internet 2, the original Internet is the same one we're using today. It has turned out to scale brilliantly because the foundation of it was so good. And of course we've spent many podcasts in the past talking about that, explaining exactly why it has managed to scale the way it has. Similarly, the bitcoin is the currency. There will always be other wannabes, like there's already Litecoin we talked about last week. But this one has made it. And...

Leo: It's possible for there to be multiple choices. Or no?

Steve: Sure. In the same way that there are...

Leo: I mean, there's multiple national, governmental currencies.

Steve: Yes, exactly. And there are people who say, oh, I don't trust the dollar anymore. I'm moving mine over into the mark or the yen or the whatever. So, yeah.

Leo: You know, I think I should set up - we have a dollar donation system from PayPal. I think I should set up a bitcoin donation system. Why not?

Steve: Well, I think that absolutely makes sense. The EFF had that for a while, the Electronic Frontier Foundation. And they liked having it because they liked the concept. But they took it down with an explanation that they felt, due to the level of their influence, they were endorsing bitcoinage. And that made them feel a little

uncomfortable. My guess is, if they still had it today, they would probably not take it down. I doubt they'll put it back up. But now with the recent decision by the Treasury Department saying, yes, this is legal, folks, it's one of the things that really - I think that was the catalyst for what we've seen in the last month.

Leo: Yes, yes. It also established kind of a standard. Some don't like the idea of any governmental regulation. But at least it kind of said, you know, we're going to treat these as a certain kind of entity. This is the real deal. And by doing that, I think it did totally validate Bitcoin. Totally validated it. But this is not a show about Bitcoin, and I've just - I've derailed you.

Steve: Well, what we've been talking about is a distributed currency. And today's podcast is about the technology of distributed databases. And it's unfortunate that the way of indexing the database has become the way they are known because the way you index them is through what's called a "hash." And so the technology for finding the data in the cloud is called a "distributed hash table." So today's topic for the podcast is Distributed Hash Tables. But that's a lot less dry than it sounds because this is actually - it's the fundamental solution which has been worked out for how you distribute a database on the Internet in a way that is robust, meaning that it's reliable and available. And it is tolerant of pieces of the database coming and going at will, yet it all still works. So there's redundancy built in. It is also impervious to tampering by authorities. The original centralized database that Napster used for its audio filesharing was vulnerable because the authorities could stomp on the Napster hub.

Leo: Centralized.

Steve: It was centralized. BitTorrent does use this technology that we'll talk about today, distributed hash tables and a distributed database.

Leo: Oh. So it is germane. That's good.

Steve: Oh, yeah, yeah, yeah, yeah. I mean, it is in use now. We know that the Tor network, which is what put us onto this topic, is using it as their own directory and index because there is no single point of failure. There's no one anyone can - any authority who wants to control it can grab in order to shut it down. And even Amazon, it turns out, uses this technology for all of their big storage. This is how Amazon stores things because they need high availability, redundancy. They're inherently multi-datacenter cloud-based. But what they looked at when they looked at their needs, they said, you know, SQL, the Structured Query Language, SQL, is really too heavy for us. We don't need all of that kind of stuff.

When you think about going to Amazon.com, a web page is being assembled from a whole bunch of different stuff, is all being pulled together onto the page. Well, it turns out that all of those little bits and pieces have tags, and somewhere there is an image, there is a blob of text, there's a this or a that, that is, that you find with that tag. So a page on Amazon.com is just a list of tags. And they use a distributed hash table, distributed database in order to instantly yank that together. And we see every time we play with Amazon that it works.

Leo: Steve Gibson at GRC.com. His Twitter handle is @SGgrc. There's some good news, for Cox customers, anyway.

Steve: Yes. Well, what I like about this, and I didn't know about this, is that Cox - which is a major cable provider around the country, it happens to be my cable modem provider and others in Southern California - they have taken proactive responsibility for blocking the Universal Plug & Play port 1900 that we've talked about. We've talked about the danger of an ISP blocking it because it uses the UDP protocol, which DNS also uses. It might, if it was blocking it with sort of a heavy hand, it might also cause some DNS problems. But technically, even those would be very transient, if anyone even noticed the problem at all.

And if users are using Cox's DNS, which was what you get when you sign up with Cox, unless, for example, you deliberately used OpenDNS or Google's DNS or override the normal DNS settings, then you wouldn't even have that problem because your DNS queries would not be attempting to leave the ISP and come back, where they might get blocked if they happened to have left from your port 1900 and they were trying to come back in to your port 1900. That's the action that a block of port 1900 would close. Someone found a page on Cox's website that shows a complete list, not of just UPnP that they're blocking, but everything. And it's sort of nice to know that they've...

Leo: No, that's a good thing to know.

Steve: Yeah. In fact, Leo, if you click that link in the show notes, you can see that they're blocking port 25 on TCP. They show the protocol is SMTP, of course, which we know, and they're saying they're blocking it to prevent SMTP relays, so to prevent their customers from relaying spam, if they have that.

Leo: That's very common. Most ISPs do that.

Steve: Yes. And they're blocking 80 so that...

Leo: What?

Steve: Because they don't want...

Leo: But that's...

Steve: Well, yep, I know.

Leo: That's because that's the inbound web traffic. So you could send a web request and establish a conversation, but nobody can send one to you. That means you can't run a web server.

Steve: Exactly. Exactly. Because when we initiate requests, we clients, clients begin issuing requests from port 1024 and upwards. It technically can go all the way up to 65535, but normally it wraps around about 5000 for most purposes. So they are blocking inbound port 80 to prevent people from running web services because that's prone to abuse.

Leo: And it's good for business.

Steve: And that's just not, yeah, it's not - because of the service agreement with them, exactly.

Leo: Right. They say you can't run a web server, so don't try.

Steve: Exactly.

Leo: And they'll block it, yeah.

Steve: And then they're blocking - they're famously blocking the traditional problematic ports for Microsoft, 135 through 139, and they talk about that. Also 445, which is the updated Windows printer and filesharing port for Microsoft. They're also blocking the SQL port, which makes sense, 1433 and 1434, just because for some reason many things you install these days will, like, bring a SQL server with them because they want to use it locally. But in typical not-planning-ahead fashion, the SQL server advertises its services on that port. And if you were not behind a router, then that port would accept SQL queries technically from anyone in the public Internet, which is really not what you probably had in mind. So it makes sense that they would block that, too, basically protecting users from themselves.

And then finally port 1900. And that's of course the Universal Plug & Play port. What's interesting is that the person who posted this, or shared this, I don't remember now if it was through Twitter or on GRC's own newsgroups, I asked, I said, I wonder how long this has been going on? And they got back to me and said they asked a Cox person, who didn't really feel like they had a definitive answer. But they said "forever." So, meaning at least not just in reaction to this recent report of the millions of exposed Universal Plug & Play ports on the Internet. But that's something that they, if we're to believe them, preemptively recognized as a problem and took care of ahead of time. So I say bravo to them. And very nice that they're publishing that. I don't know what other ISPs do or don't, but it's nice that Cox is.

One thing I do know that an ISP is doing that I wish were not the case is that Comcast...

Leo: Oh, my ISP.

Steve: ...has recently been found to be injecting, not only injecting JavaScript into the pages that people browse to, that is, so you go to any website through Comcast as your ISP, and when the page at Google or MSN or anything, Facebook, is returned to you, Comcast is using deep packet inspection technology to inject their own JavaScript into

the page from the remote site. And, I mean, that's annoying sort of just from a purist standpoint. But, moreover, it's badly written JavaScript.

So, for example, one of the problems that JavaScript has is it's trying to - it's still going through growing pains. We're at ECMAScript - which is like the formal name of JavaScript - we're aiming toward version 6. And they're adding features to mature the language. But one of the original concepts was, oh, anybody can write JavaScript. Well, apparently anyone did. This is very poorly written. And, for example, unless you take extreme measures, all the variables in JavaScript are global, so that you have to be very careful when you are mixing canned JavaScript that there isn't a collision of variable names. If there is...

Leo: It's so annoying.

Steve: I know. Then those various scripts that intended for these variables to be private to themselves end up sharing the value, so they're stomping all over each other. Well, this JavaScript does that. It has not been written so that it uses containment. There's a technology called "closure" that existing JavaScript can use which is so complicated that it's got its own book. There's a book that O'Reilly produces on JavaScript closures. And it's on my "when I really have extra time to get to it." I've used closures for my own code because I want to be careful about not doing this. There were no closures used here. So any variable names in the script that you have no control over, that the site you're visiting has no knowledge is being added to its pages it's sending to you, is potentially stomping all over the variables that they created. And it's doing other annoying things.

The idea is that this is Comcast's means of informing you when you've reached 90 percent of their bandwidth allocation for you for the month. So if you're going along, surfing the web, doing your stuff, at some point, hopefully very near the end of the month, up will come a popup on your browser from Comcast saying, notice, you have exceeded 90 percent of your monthly bandwidth allocation. All of this...

Leo: So that's the purpose of this?

Steve: Yes.

Leo: But it doesn't happen until you've reached that? Or is it happening to everybody?

Steve: No.

Leo: Oh, okay.

Steve: Apparently it is not being injected until they need to give you the notice. They're using old code. They're using code they took from other sites whose licenses they're apparently violating.

Leo: Oh, god. Some moron did this.

Steve: Yes. This was really...

Leo: Probably a contractor. Oh, I can do that for you, no problem.

Steve: Yeah, uh-huh, yeah.

Leo: Yeah, I can set that up.

Steve: And then, while the page is there, even after you close the dialogue, there are - AJAX is a technology in HTML5 and beyond that allows the browser to initiate queries out to the hosting site. And you're prevented, AJAX is prevented from sending queries anywhere else for security's sake. You wouldn't want your browser to be off doing other things to other sites. So they've got code in there such that every page with this script starts making queries back to the host every five seconds. So if you're someone like me, who has lots of pages open, if Comcast were my ISP - they're not, but were they - then every one of those pages would have had the same script redundantly loaded into it, and they would all be generating outbound queries every five seconds. So it's just bad.

Leo: Stupid, stupid, stupid, stupid. Stupid.

Steve: It really is bad.

Leo: Gosh darn them.

Steve: Yeah. What they should do is, if this is what they want to do, they ought to, at the beginning of a session where your bandwidth is exceeded, just bring up an intercept page. We're all used to that when we go to Starbucks or any standard - like an airport where they say - your normal use is intercepted by a page that says agree to our terms of service if you want to use our free bandwidth, and disclaimers and all that yada yada. And so they go, okay, yeah. That's all they would have to do, just say, wait, stop, you're at 90 percent, wanted to let you know. Click here for more information, click here to lift the ban, whatever. And then use the Internet in an unmolested fashion.

This is a case, a further case for encryption, HTTP, because that they cannot intercept. So this is another reason why the whole notion of us, the 'Net moving to SSL/TLS all the time is a good thing. It prevents our ISP from monitoring what we're doing, seeing what's going on, and from modifying the pages that we receive on the fly. Definitely an annoying thing.

Leo: We've seen that kind of stuff before with, like, changing 404s into advertising and stuff like that. But this is really terrible.

Steve: Okay. Now, you've got to go to a page, Leo, that bit.ly link that I show. I'll just say it to you, and you can click...

Leo: Yeah, go ahead, [indiscernible], okay, now you can say it. I mean, you can say it out loud if you want.

Steve: Yes, and I will. It is - I tweeted...

Leo: We don't want it to bring down the site before I get to it.

Steve: Before. But for everyone who's interested, I created a shortcut to an otherwise very long URL, bit.ly/javaage, all lowercase, j-a-v-a-a-g-e. Now, Leo, the right way to look at this, because it's a little confusing at first, is at 12:00 o'clock, 12:00 o'clock is sort of the start and the end of this pie chart. And so as we go clockwise from the dark side to the lighter side of the colors, we are moving back in history. This is a...

Leo: It's a pie chart of versions of Java that was created by Websense.

Steve: Currently in use. That's the key.

Leo: And the biggest pie slice is October 2009, with 9 percent compared to 5 percent of the current version. And 8.5 percent for the original Java 1.0. Jiminy.

Steve: Now, how many, you know, there are pages tracking the days since the latest exploit. Everyone, I mean, one of the things we spend an inordinate amount of time on because unfortunately it's important, is the incredible insecurity of Java. And they are constantly producing emergency releases in order to fix the latest problem. What this says is this is a population study based on the current version loaded into the browsers on the Internet. So these are the versions of Java that are accessible for exploitation.

And looking at this chart, again, bit.ly/javaage, you have to just despair because it says that this stuff is not being updated. I mean, this is where the exploits are coming from today more than anywhere else. We're on Episode, what, 398 of Security Now!. Back in the 100s we were talking about email spam, you know, don't click on the links in email. And while that's certainly still true, this has all moved to the 'Net and to these vulnerable plugins, Java, Flash, and Acrobat, or Adobe Reader. And so here we're seeing the truth of the fact that old versions never die, and they absolutely should.

Now, the problem is that only the new versions auto update. So if you have an old version, it never had auto update technology. I mean, even though arguably two years ago, as you said, Leo, the October '09, what, no, that's more than two years ago, that's four years ago. But we've had Windows Update and automatic updates and updates, and look how many prior versions of Java there were before that? That's v1.6.16. They started at 1.0. There were already tons of major and minor updates, yet no automatic update facility. They were in denial that, like, oh, this will be the last bug. No. Then, we found the last bug. This is, you know, this is it. This was four years ago. Oh, no. This was the last bug. We don't have to update this ever again. Uh-huh.

So the problem is, these Javas will never change. I mean, unless something somehow went through and swept them. Now, I wonder how they're being invoked because certainly Firefox and Chrome are taking responsibility. I guess IE - IE's taking responsibility for Flash now, but I don't think it does anything to warn you about antique Java clones, I mean Java versions, in IE. And so my feeling is, since Java itself cannot, it is not taking responsibility for keeping itself current back before that technology was added, it's up to the hosts of the Java plugin to do so.

And we really need all browsers to be proactive about this and just stop allowing these noncurrent versions to function. It's a trivial thing for them to do, to check the release version of the Java plugin which is trying to run in them and say, uh, wait a minute. And we know that Firefox does that. And I'm pretty sure that Chrome must be doing that, too, because they've been so proactive.

And as a consequence, at the moment, vulnerable version, the most recent release is 1.7.15. And we already know - and the similar release on the v1.6 track is 1.6.41 - 93.77 percent, almost 94 percent of all browsers today are vulnerable to that exploit and all the previous ones.

Leo: Nice.

Steve: Yeah. It's just - really, really, really you want to stay away from this. Just be very careful of Java.

Leo: Now, I have created a bitcoin QR code that I will now flash on the screen. If anybody is so hardcore that they want to donate bitcoin to TWiT, this will go to our TWiT account.

Steve: I think that makes a lot of sense, Leo.

Leo: We'll put it on the website. You know, I'll be very curious as to how much...

Steve: Be a great survey, just to receive one.

Leo: Yeah. And, you know, even if we get a couple of coins, that's some money, isn't it.

Steve: Well, but remember, they can send you 0.0000000...

Leo: And this QR code, you can, when you set it up, specify, I'd like one bitcoin. I didn't say. Donate whatever fraction thereof. Send us a bitpenny, or whatever the hell they're called.

Steve: I think we're going to see hoarding going on because I'm sure not doing anything

with my 50.

Leo: I ain't spending any.

Steve: I'm sitting on mine, yeah.

Leo: Yeah, I ain't spending any. So if you have Bitcoin on your smartphone, you can actually use this to send bitcoin. I'm told. I have no idea. Now, I couldn't find - I had created a bitcoin - in 2010 somebody emailed us, emailed me and said, hey, I'd like to give you a bitcoin - I wish I'd taken it now - if you have a bitcoin email address. And I said, what's bitcoin? So I set up a bitcoin - I was even doing some mining. I wasn't as lucky as you. Now it's not even worth doing it, right, you'd have to set up such a high-end machine. But I couldn't find that old address. Now, the key on this, right, is to back it up because that's your wallet, and that's it. If you - your system's hard drive crashed, and I had a bunch of bitcoin in it, I'm in deep doodoo.

Steve: Yes. Yes. It's purely bits.

Leo: It's just bits.

Steve: It's purely electronic.

Leo: Okay. So as soon as I have any coin in my wallet - I guess I should back up the number, whatever this is. I don't even know what this is.

Steve: Yes, well, when we lose our checkbook it's okay because the bank is actually still holding our money.

Leo: Yes. There ain't no bank.

Steve: Not the case.

Leo: No bank here. Sorry.

Steve: So, no, this is good. So something's disturbing.

Leo: I'm thinking of yabba-dabba-do. So every time a bitcoin is donated, a yabba-dabba-do.

Steve: Oh, I'll send you my WAV file.

Leo: Yeah. Go ahead, I'm sorry.

Steve: So something disturbing is going on with Apache.

Leo: Uh-oh.

Steve: It was first seen last summer, August of 2012. And it's becoming prevalent and now worrisome. There have been several security researchers who've been following this. And, for example, there was an infection of the L.A. Times, which we briefly mentioned last month, or, I'm sorry, February. Seagate's media site, media.seagate.com, got infected last month, in March. The symptom is that IFrames, which are "I" for "inline," inline frames are being conditionally injected into the pages being served by these infected Apache servers. Every server that has been found that's been infected is running version of Apache v2.2.2 or later. So that's one common factor.

The disturbing thing is no one knows how they are being - how these Apache servers are being infected. But more than 2,000 servers have been. And those 2,000 servers are hosting, because many of them are hosting multiple sites, virtual sites, they are hosting more than 20,000 websites. So 20,000 websites, hosted by 2,000 web servers - and these numbers, of course, are rough, and it's difficult to find these, I'll explain why in a second - are infected. And what's making this so difficult for investigators is what has been found in infected servers is a malicious Apache module. So it's one module. Apache is inherently module.

In the HTTP config file you have a whole batch of load module statements which, when the Apache service starts up, it reads the file to figure out what it's supposed to be doing, how it's supposed to be acting as a server. And it sucks in all of these different modules, PHP and so forth, for offering the services that it needs. One of these modules is malicious on these servers. And it's in the so-called "Apache pipeline," which is the term they use for the request processing, the idea being that a module in the request pipeline is able to do anything it wants with every request made to the server. So, and it has all the request headers that it's able to see.

So what the developers of this module have done is they have given it knowledge to thwart analysis. It will never inject the malicious IFrame into the IP of any known security researcher. So security researchers can't scan the 'Net. They can't go to suspected sites. When they go there, the site looks fine. Somebody else goes there from a different IP, and it's not fine. But way more than that. They look at the user agent, which remember in the request headers is the claim that the browser makes about its heritage - is it Safari, is it Firefox, is it IE, and so forth. So they're only selecting, currently, they're only targeting specific operating systems because one of the things in the user agent is typically this is the browser I am, and this is the OS I'm running on. They're only targeting Windows for these particular attacks.

Now, again, the infected server could be running, and is typically running on Linux sites or on Linux OS, hosting Apache, which is then hosting either one to hundreds of websites. But, for example, they blacklist search engine spiders. So the search engines won't pick up the IFrame. Again, as we've talked about before, that's been a very clever means of finding problems. Remember, for example, that webcams were being indexed. So you could just do a search and find all the webcams because they all had a particular user agent in their reply, and so anyone who was spidering the 'Net and collecting user

agent strings could say, oh, here's all the webcams that you can log onto remotely without a password.

But nope, these guys don't do that. They deliberately don't inject this IFrame for search engines. They also use cookies to manage return visitors, and they look at their referrer header. Remember the referrer tells a website where this query came from, where the user was when they clicked on something. And so, for example, if you receive the URL of a suspect site, if you were a researcher, and say that you were using different IPs on the theory that maybe it's IP sensitive, which it is. But so you were going to go to a different IP so that that wouldn't catch it. And you enter the URL into the address bar. Well, there's no referrer, so it'll be blank. And this thing is smart enough not to inject. It only injects if it knows that the link on a given search engine is valid for this site, and that means you're probably clicking the link, although it's spoofable. But still, it's another level of filter.

So you can see that what we have - and this is why this has been sort of pernicious and persistent and slowly growing over time. What it does, the IFrame refers people's browsers to additional malicious content that are on otherwise valid sites. So other sites are infected with the actual malware that's going to infect your browser. This whole system I've been talking about is the way of getting you essentially to visit a site you're not really visiting. So if a bad site were infected, it's like, okay, if people go there, they'll get infected. But if they don't go there, they won't. Instead, other good sites, 20,000 of them, are now injecting an IFrame which induces your browser to go to the bad site to get itself infected. And so this is sort of a delivery, sort of a second-order delivery system for that.

One of the things that's also been found on these compromised Apache servers is that the SSH daemon, the secure shell service, has been compromised. A backdoor has been installed in every one of these machines that has been seen that allows remote attackers to access the machine, bypassing the regular authentication. Thus, that's why you would want a compromised secure shell, so that you don't need to log on with real username and password and whatever other credentials.

But it does much more than that. It captures any valid credentials which are used to log on and sends them off to a repository somewhere so that the attackers, having once compromised this machine, then get the valid credentials. And what's often the case is that the same credentials are used on many different systems by common admins. So this gives them a dictionary of username and passwords to try elsewhere to get in. And so this thing tends to spread.

So they're monitoring anyone's login, that is, admin login. And if anyone uses that login somewhere else - oh, and I guess the other thing that happens is it's often the case that someone will log into the system and then will secure shell from there to another system. And that they capture, as well. So they're able to monitor anyone using a compromised system to access and administer another not-yet-compromised system, in the process getting access to it, which is, again, the way this tends to spread. So it is daunting. It's known to be going on with Apache systems. There are a couple researchers who have been tracking it. There's a great blog called UnmaskParasites.com. And Dennis, who's the - who runs that site, Unmask Parasites, has a blog. And he wrote of this. He said, "While this hack is not new, I could not find reliable information about how hackers break into servers and get root access. The malicious modules are owned by root, and httpd.conf files can only be modified by root." So that tells him, for sure, that somehow someone's getting root on these servers.

He said that, "At this point I can say that it doesn't look like a security issue of some

control panel." Remember we've seen that in the past. He said, "The two infected sites I worked with had different control panels," Virtualmin on one and cPanel on another. He says, "The malicious module doesn't come from Linux repositories," so that's not how it's getting into these machines. He said, "So someone somehow breaks into servers with root permissions, which is quite alarming. One of the servers had only one user" - it was a dedicated server - "used strong passwords, didn't allow remote root logins, used custom ports, and Fail2ban to prevent brute-force attack. Nonetheless, it was hacked within one month, and its administrator could not find signs of the intrusion in log files, only legitimate logins." And he says, "Of course root access gives you the ability to remove most possible traces."

So we'll keep our eye on this. People running Apache v2.2.2 and older may want to check out UnmaskParasites.com. There's much more there that's more specific to administering the site. He gives a lot of scripts and specific things you can do to verify that the modules your system is currently using have not been modified and that no additional unknown ones have been added. And it would certainly seem to be something worth doing because at this point it looks like there is an unknown means for root access, somehow gaining root access to servers. And the people who are taking advantage of this are being so careful not to let this get found because they recognize this is the golden goose for them, and they don't want it to get loose and be closed. So it looks like someone's taking very careful advantage of a currently unknown means of compromising Linux servers running this version of Apache, we don't know if it's through Linux or through Apache, and through where? But we'll keep an eye on it.

Leo: Yeah, that's how our servers got hacked was not through Drupal, but through modules, older modules that had been installed on top of the Drupal. It's hard to keep track. You have all these modules doing things. And so you don't just have to watch your main code base, but all these little modules that are in there.

Steve: Yeah, have you looked at how quickly any apps in iOS get old? It's ridiculous. I mean, I...

Leo: Yeah, they're updated all the time, yeah.

Steve: I'm being punished now for having played with so many of them. It's like, oh, you've got 56 updates. It's like, oh, my god.

Leo: Yeah, I know. You've got to do it every day. Hey, I was running the Bitcoin QT software, which is kind of the default.

Steve: That's the one, yup.

Leo: Yeah, kind of basic software. And I was going into the settings, and I noticed this: Map port using UPnP. And that's checked by default. Now it seems to be connecting, so maybe you don't have to. Maybe it doesn't need the special port. But I thought that was kind of interesting.

Steve: Anything that wants to be part of a network. And Bitcoin needs to communicate with many other nodes in order to update itself, to bring itself current with the current hash history. And then the idea is that, when it succeeds in minting a coin, it sends that news out, and it requires that other nodes confirm that it found the hash. And so it's that communal agreement that runs the whole system. So you cannot be a bitcoin miner on an island.

Leo: And offline, yeah, yeah.

Steve: You've got to be part of the network. And you're also, when someone says, hey, I got one, it's like, oh, crap. Okay, let me see, and, you know...

Leo: And now what do I do; right? Yeah. But we did get, apparently, somebody in the chatroom has given us, at least I haven't received it yet...

[Loud yabba-dabba do]

Leo: Our first bitcoin has appeared, apparently, in the wallet. So...

Steve: And that sounds like the right file. Very, very familiar to me.

Leo: So there you go. Apparently I don't need UPnP unless I'm getting incoming connections. I don't know what that means. But I'm making outbound connections, obviously, to the Bitcoin servers.

Steve: Good, good.

Leo: So that's all well and good. And I've encrypted my wallet, and I'm backing it up. I wish it said where the wallet was stored, at least on the Mac, and it's not immediately obvious. I'll have to dig around because I'd like to put the wallet on a Dropbox, and then I wouldn't have to back it up; right? As long as it's encrypted, having it in a Dropbox would mean it's on a bunch of machines. It'd be pretty safe that way.

Steve: Yes. And in fact I know that on Windows it's under appdata/bitcoin, and then it's like /wallet. And it's a little - it's a collection of files that form the wallet.

Leo: Can I double-click? It'd be nice if I could double-click the wallet and start the Bitcoin that way, and then that way I could start it from any location. I'll have to - I'll play around with it.

Steve: Yeah.

Leo: But, yeah, it's nice, isn't it?

Steve: It's real.

Leo: I'm already $50, what is it, no, more than $100 richer.

Steve: You're kidding.

Leo: Of course, I can't do anything with it.

Steve: Wow.

Leo: Well, somebody gave me a bitcoin. That's a hundred, what is it, 113 bucks.

Steve: Nice.

Leo: Very generous.

Steve: I'll give you a podcast instead.

Leo: You have 50 bitcoins. You could - you could - you could - we are going to put the bitcoin address, which is just a long crypto number, and the QR code to make it easier for you, so you don't have to enter it by hand on our website, if you want to donate bitcoins. Thank you. And thank you to - now, I forgot his name, it's DarkHaven or something like that, for our first bitcoin.

Steve: Nice.

Leo: Yeah.

Steve: So I have an announcement.

Leo: Uh-oh. Are you getting married?

Steve: No, Leo.

Leo: Having a baby?

Steve: Mmm...

Leo: Okay.

Steve: Who's laughing in the background? Somebody who probably is going to have a baby one of these days.

Leo: Maybe. It's our newlyweds in the room from Wisconsin.

Steve: So GRC is announcing today a new service.

Leo: Whoa. You are prolific all of a sudden.

Steve: I've been cranking away, baby. I referred to this once, and it is now up and public: GRC.com/fingerprints. It's under the main menu under Services because it's a new service from GRC. This is the thing that I talked about a few months ago. It took me literally until yesterday to get it all nailed down, working perfectly. This allows people to detect when somebody is intercepting their SSL, their TLS, their HTTPS connections. And I'm really proud of it. It will be here forever. Of course it's free, like everything here at GRC. And I think it will end up probably over time becoming popular.

The idea is that, if you are in a corporation that is secretly decrypting your SSL traffic, and you're connecting to the corporate server; and then it's decrypting your traffic and then inspecting it, filtering it, logging it, whatever; and then it's connecting to the secure service that you think you're connecting to, it's spoofing the certificate that you receive. Well, it is impossible, thank goodness, to perfectly spoof a certificate from a remote server because the certificate contains, as we know, the remote server's public key. And the public key matches the private key, and the private key is secret. So nobody knows, for example, Google's private key. Believe me, they're not letting anybody find that out because that would be the end of the world. So that means that the authentic certificate from Google contains the public key, which we use for encrypting data to Google.

But a certificate which is spoofing Google, pretending to be Google, has to be offering us a public key that it says is from Google. But it can't be the same public key as Google's because we don't know, because the person spoofing doesn't know Google's private key. So that means that it is impossible for a certificate to have the same hash. And so what all browsers allow you to do is to view the details of the certificate you have from a site. For example, if you're looking at GRC.com, that page, GRC.com/fingerprints, will only allow itself to be displayed over a secure connection. And so you can always right-click on it and do the View Certificate.

And all browsers, and on that page I give instructions for all the different popular browsers, step by step, here's where you can find the data, although the browsers don't hide it. Normally it's right there, and it says "thumbprint" or "fingerprint." And it's a series of 20 hex characters which is the - it's the hash that the browser just made of the certificate. The certificate contains other internal hashes for their own purposes. But this is the hash of the certificate that the browser calculated to say this is the fingerprint of the certificate. No one's ever really done anything with this before until now.

So the idea is GRC will make a connection for you. I display a set of 11 different server certificates, just sort of the popular ones, Twitter and Facebook and WordPress and Tumblr and so forth, just sort of as a reference. But so the idea is that GRC's servers are right on the Internet backbone. Level 3 is a Tier 1 provider. Nobody is filtering my connection. So I create the connections and get the security certificates from those servers, or you can also enter your own custom one, whatever particular one you may care about, like your bank. And so I will show you the authentic fingerprint of the real certificate. And then you simply compare it with the one your browser shows you for the same site. And if they're the same, you absolutely know, absolutely, that there is no interception going on. And if they're different, then you need to be careful and maybe worry a little bit because, if you didn't know this was happening, then it raises some flags.

So I expect this to be a long-running and very useful service. Anyone, anywhere, can instantly check, get from me, using this page, the authentic fingerprint of any site they wish and then compare it easily with the fingerprint that their browser shows them for the same site. And if there's been a man in the middle attack, whether malicious or corporate or educational or your church or whatever organization, this is happening more and more because, as we've talked about, more and more sites are using SSL all the time for the sake of privacy and security. But that's making people uncomfortable, that is, people who want to monitor and filter your traffic.

There's even a reference that gives you a sense for this on that page. I found a blog posting a couple months ago from one of the companies that is doing this kind of - offering this service as an appliance, where they say that they've just now removed the LGBT filtering checkbox from their user interface in reaction to people complaining, apparently, that they were explicitly offering LGBT filtering and tracking of people using their service. No, I'm not kidding. It's bad, Leo.

Leo: That sucks, yeah.

Steve: Yeah. And also a dialogue there from the Windows, I'm sorry, the Microsoft product, which is a "Forefront Threat Gateway," they call it, which makes it very clear that they're "inspecting," as they call it, they're inspecting your HTTPS traffic.

Leo: Oh, my god.

Steve: Yeah, a little inspection. So...

Leo: "Enabling inspection may have legal implications. You should verify if using this protection is in compliance with your corporate policy." Good lord. Geez.

Steve: So we needed a way to quickly check to see if this is happening to us, and GRC offers that now. So I'm really glad I did that.

Leo: Excellent. That's really excellent.

Steve: And this came from the mailbag, by the way. I referred to this way back when. Somebody wrote, one of our listeners said, hey, Steve, is this possible? And it's like, oh, that's a good idea. I like that. And it's taken me quite a while, but it's done. So it's there and always will be.

Also, just a little update. The second most recent service offered was the addition of the Universal Plug & Play probe to ShieldsUP!. And when I looked this morning, we were at 4,060 routers identified. So it continues creeping up as more and more people find it.

Leo: 4,061 now.

Steve: Ah, good.

Leo: One more. Not mine. One more.

Steve: Also some news. We have a new Firefox. When I saw this yesterday, I went over and did, under Help and then About, and that kind of kicks it into action. It says, oh, you want to know about version. And I was on 19.0.2. And asking it about itself started it downloading a 10.3MB replacement, which it then updated. Now I'm on version 20. We have a couple new things that people will care about and a lot of little other things.

We have something new for Firefox, which is convenient, is per window private browsing. So under the File menu there's the New Tab, to give you - obviously you want to open another tab. There's New Window. And then there's New Private Window. So you are now able to just leave your existing session up and running, you don't have to shut down and restart for the whole browser to go into privacy mode. And anything you do in that window is not recorded. And there's a cute little purple Mardi Gras mask that appears in the upper right-hand corner of the window when that's the one that you're in, just to say, yep, we're protecting you. Nothing gets written to the disk, no cookies saved, no history saved, and so forth. So that's convenient.

There's also - they've updated the Download Manager so that it's got a better user interface, a better "experience," as they say. And they have furthered their technology to prevent plugins from being able to hang the browser. I haven't had that happen for me in Firefox for quite a while. It used to happen. It was annoying. But now plugins can be expressly individually closed without needing to recycle the entire browser.

Then there's the regular security fixes, performance improvements. They've added some more things to ECMAScript 6, so they're moving their support for that forward. There's a new JavaScript profiler for people who want to improve performance. And I got a kick out of this. They've added a new HTML5 feature, Get User Media is the name. And it implements web access to the user's microphone and camera.

Leo: Oh, wow.

Steve: What could go wrong with that [laughing].

Leo: A la Flash. Interesting. Well, but that's probably maybe WebRTC or something like that.

Steve: Yes. Yes. They're moving toward being able to host real-time, like, chat.

Leo: You can do that with WebRTC right now through Google. So, yeah, that's interesting.

Steve: Yeah. So that'll be a good thing. I mean, again, I'm bullish, as I mentioned when I was talking last week about the asm.js work that they're doing that allows them to get very high-performance JavaScript without breaking compatibility. I mean, I'm bullish about the browser as the future. This is the cloud access platform. So these are all good things they're doing.

Leo: And I have to say, when it's open source, I have a little more confidence than when it's closed source, written by Adobe. At least somebody's looking at that stuff.

Steve: Yeah. And can fix it quickly.

Leo: And can fix it, exactly.

Steve: Oh, and when I was - after upgrading I saw a notification that I hadn't - maybe I'd seen it before, but I hadn't mentioned it. And that is, along the bottom it said "Firefox automatically sends some data to Mozilla so that we can improve your experience." And then over on the far right was a button, says "Choose what I share." And so I thought, okay, let's see what that is. So I clicked that. And all that did was open up the standard Options dialogue under the Advanced category and the Data Choices tab. So anybody can see that anytime you want by going to the Options dialogue in Firefox, Advanced category, Data Choices tab. And there are two things there. There's telemetry, which was not checked, which says - which I respect, I'm glad they didn't - says "Shares performance, usage, hardware, and customization data about your browser with Mozilla to help us make Firefox better." And then - and I turned it on because it's like, yeah, I want them to know I'm running with 85 tabs so they can think, oh...

Leo: Right, there's somebody doing that.

Steve: We've got to handle these people.

Leo: We've got an outlier here.

Steve: We've got to be able to make that scrollable. And then the second one was Crash Reporter, which was turned on by default, which "Firefox submits crash reports to help Mozilla make your browser more stable and secure." And that was on, and I left it on. So

those things are there.

Our prolific animator has continued animating old episodes, old but not dusty, still useful.

Leo: They're all good still.

Steve: Yeah.

Leo: This is an archive of brilliance.

Steve: So AskMrWizard.com, or AskMrWizard.com/securitynow. Bob Bosen is running that show over there, and he's added Episode 11, that is Security Now! Episode 11 on Bad WiFi Security, and that's in five animated video segments; Episode 13, WPA: WiFi Security Done Right, in two video segments; and Episode 14, VPN Theory, and he's only got the first of several segments complete so far. But he's moving ahead.

Leo: That's awesome. That's great.

Steve: Doing animated video to supplement the podcast audio. So, yeah.

Leo: Wonderful. Wonderful.

Steve: I think that's really great. And I wanted to just check back in. There's been a lot of activity in Mark Thompson and company's - I was calling it Smush.

Leo: Yes. They corrected us.

Steve: Smoosh.

Leo: Smoosh. It's a Smoosh Box, yes.

Steve: I didn't want to be smooching it, so I got Smush, no, Smoosh Box [SmushBox].

Leo: By the way, I ordered one. You know.

Steve: Yes, and Mark saw and was pleased. And I don't think he'd heard the podcast at that point, but he has since. In fact, he sent out a notice to all Kickstarter...

Leo: They're so close, 15 days. They're just 3,500 away from their goal.

Steve: Yeah, so they've got 16.4 something; right?

Leo: Yeah, 16,446 out of 20,000 they want to raise. So we're getting close to the Smoosh Box.

Steve: Yeah, and frankly, I don't think - this is not a do or die thing. It's not like they're not going to make it if they don't...

Leo: Don't say that.

Steve: Okay.

Leo: Don't say that. All the Smush Kit early backers are gone now. They still have 35 of the $250 later backers available, and...

Steve: Well, and, now, that's significant because I got some feedback from our listeners. Several people sent me links to services. And so it's worth mentioning there are definitely services, there's actually many...

Leo: Oh, yeah, will do this.

Steve: ...that allow you to, yes, allow you to sign up. But what happened is Mark, for his purposes, was using a service that sold the phone numbers.

Leo: Right.

Steve: And so Mark said, okay, clearly this doesn't work.

Leo: Yeah. You want to run your own service.

Steve: And that's the point is, yes. And I've looked at pricing. And they, of course, it's like for a while I thought, oh, look, I could put my website, give it to Amazon. It's like, oh, no, no, no, no. When you start doing the math, it's not feasible because they want to make money. And of course these services want to make money. They've having to pay something on the back end, so they're marking it up.

Now, one person sent me a link to hardware that their company uses. Multi-Tech has something called the SF, Sam Fox, hyphen 100G. And it is essentially what this box does. So Multi-Tech has it, called the SF100-G. I thought, oh, let's go look. So Beach Audio has it for $388.46. Gemini Computers at 374. PLC Center for 451. I don't think they're going to sell many of those at that price. And eBay sold one just two months ago, middle of February, on the 19th, for $390.

So there does exist something that looks like this. I have no idea in detail how it compares to what Mark and his group are putting together. But we're getting it for more than $100 less than the best price here, when we get it for 250 bucks. So I'm glad these guys have done something, and I think it'll be a long-term win for them. And I'm excited to play with this thing.

Leo: That's really cool. Yeah, I am, too. What we're going to do, so people understand, what I'm planning on doing, and I've already talked to my web team, and they seem to think it's doable, is set up a checklist, just like you would subscribe to a newsletter, but you could subscribe to text messages when Leo is in the building, the podcast could be beginning any minute now, the podcast actually began...

Steve: Hey, I - I need that, yeah.

Leo: The podcast is done, and the podcast is produced and available for download, that kind of thing.

Steve: I love the idea of getting texts when it has been posted online. That's great, yeah.

Leo: Yeah. So I'm trying to think, now, there are 25 shows. There's probably two texts per show, "started recording" and "is available for download," and maybe one more on my shows where I'm actually - the preshow has begun. So let's say three. That's 75 different possibilities. It's already way out of control.

Steve: Yeah, so it's 25 lines and three columns.

Leo: Yeah. Well, we'll put it on each individual page. So if you go to TWiT.tv/sn there'll be a checkbox that says send me a text. Now, we will, I promise, we will not harvest the phone numbers. I don't want your cell phone number. I don't want your stinking cell phone number. And because we're doing it ourselves we can make that promise. Nobody's going to see that number. Except T-Mobile, I guess. You have to use them as a service. And then, I guess, if a thousand, 2,000, I mean, I don't know how many we might be sending out. What if 20,000 people say we want to know when Security Now! is out? We'll be sending 20,000 text messages.

Steve: Well, it's a flat rate, 25 bucks a month with T-Mobile, so...

Leo: I think it's cool.

Steve: I think it's very cool, yes.

Leo: Yeah. And we might have some emergency messages that you - you know, when my salad is disappeared, things like that, that people...

Steve: Or if you have a power failure at...

Leo: There you go. But then the SmushBox will also be dead. So that's not going to work.

Steve: Oh, yeah. Yeah, yeah.

Leo: We'll figure it out.

Steve: Put it on a UPS. Put it on a UPS.

Leo: Yeah, we'll take some votes for...

Steve: The cool, see, that's the other cool thing is that, if you use a service and your network connectivity goes down, you're screwed. Not if you use a SmushBox.

Leo: Right. Right.

Steve: Because it doesn't use a third-party outside provider. It is its own SMS texting provider.

Leo: Actually, we put that on a UPS, and unless T-Mobile's towers are down, and they're just right outside the door here, we might be, even with a power outage, able to send a text out.

Steve: Yeah. "We're still here."

Leo: It doesn't receive text in any way.

Steve: Yes. Bidirectional. Yeah, so you can do all kinds of things. You can have contests. You can have special events.

Leo: Oooh.

Steve: You can have text this code to this...

Leo: Oooh.

Steve: Yes, yes.

Leo: Oooh. That's intriguing.

Steve: Yeah. I'm going to use it to turn on my air conditioning.

Leo: Oh, I can't wait till SmushBox. Well, I hope - it looks like they might make it. They're only halfway through, and they're almost all the way funded. So that's good.

Steve: Yeah. That just means all we need to do is spread the word because we want people who could use this to know about it. That's the thing that's annoying about Kickstarter is when you show up too late, it's like, unh, can't - I wish I knew about this.

Leo: You know what's funny, they're painting a beautiful mural on the side of the building over here. And they had a Kickstarter project to raise the money for it. I didn't even know about it. Petaluma's largest mural, here we are across the street, didn't even know about it. I would have publicized it. Didn't even know about it.

Steve: I found a really fun - this is completely off topic. But ThinkGeek on Monday, a special day for us all, and finally it's not Monday so we can actually believe something that we see, yeah.

Leo: Thank god. The worst day on the Internet. I hate April Fools.

Steve: I know.

Leo: Hate it.

Steve: So they were offering something so cool: a $49 Play-Doh 3D printer.

Leo: [Laughing] I used to have one of those when I was a kid. You push the lever, and it extrudes the Play-Doh.

Steve: Well, this thing - and that, I'm sure, was the inspiration. I tweeted the link to it. If you look in my Twitter stream, you go to, what is it, bit.ly/sggrc, and that will find the link that I posted: thinkgeek.com/product/f487. I shared it on Monday because it was so well done. I mean, it showed you the machine, takes two "D" cells, or maybe it was "C" cells. It's got a little crank. And it shows you, like it's printing a droid. So it's a 3D printer, absolutely convincing-looking.

Leo: [Laughing] It's so cute.

Steve: And it runs - oh, and you hook it to an iPad. So your iPad is your - and there's free download software for your iPad.

Leo: By the way, Steve, this is almost certainly an April Fool's joke.

Steve: No, Leo. No. I've got one on order.

Leo: Really?

Steve: Yeah. Click to buy it.

[Talking simultaneously]

Leo: ...Play-Doh folks. No, this isn't - no. It's a joke.

Steve: Click the Buy It button. Tell them you want one. Because don't tell me you don't.

Leo: I want one. Continue on, Steve. I also want the - oh, look at this, Bare Paint Conductive Paint Kits. That's cool. The Moving Hand - Muscle Wire Moving Hand Kit. The Auto Bacon T-shirt. There's so many great things on here.

Steve: Yeah, they're good guys.

Leo: Think Geek. Love 'em.

Steve: So because it had a science fiction basis...

Leo: Yes.

Steve: ...I went with Jenny on Friday to see "The Host."

Leo: Is this a movie?

Steve: Yeah.

Leo: Okay.

Steve: Well, kinda. It was so bad, Leo. Oh, goodness.

Leo: Terrible, huh?

Steve: I've never put a review up on IMDB until now.

Leo: That bad.

Steve: It was that bad. And, yeah. So, and I'm just warning our listeners, I care about you all...

Leo: Don't go see it, huh?

Steve: Do not. Do not see it.

Leo: Here's one review that says, "It wasn't awful."

Steve: Keep reading. Scroll down a little bit longer.

Leo: Wow. "'This movie is unimaginably awful. Save your money,' Steve Gibson from the United States writes." It was that bad. Did you walk out?

Steve: No. Jenny kept offering that we could walk out. I was a little uncomfortable, feeling that maybe, if other people were actually enjoying it, they would think, why don't those two just leave, because we were groaning so much. Oh, gosh. It was just, I mean, the special effects were okay. But the screen - the director wrote the screen - did the screenwriting, which is never a good idea. And so...

Leo: It's from Stephenie Meyer, the author of the Twilight Saga.

Steve: Yeah.

Leo: "Choose to believe; choose to fight; choose to love." Yeah.

Steve: I think that was, you know, yeah. So anyway, don't go. Oh, and Leo, boy have we got stuff coming. We're going to have a wonderful summer. And I have a guilty pleasure that I will probably afflict myself with on Friday, which is the next installment of

"The Evil Dead" movies.

Leo: You love those? I've never seen any of them.

Steve: Oh, gosh, yes.

Leo: I should watch those.

Steve: That's why I said it was a guilty pleasure. You probably ought to watch them. I mean, it's a cult win. They are really funny. I mean, they're gory, but with a serious tongue-in-cheek, I mean, like "Army of Darkness." Did you not see "Army of Darkness"?

Leo: No.

Steve: Oh, Leo. Okay, well...

Leo: Does that involve zombies? Because I'm not a fan.

Steve: No, no. Well, it wasn't zombies. But it's really campy, tongue-in-cheek, over the top. And Sam Raimi was the producer, and Bruce Campbell starred in all...

Leo: I love Sam Raimi. He's wonderful.

Steve: And Bruce Campbell starred in all three of the first ones. He's not in this one. And I'm looking forward to seeing it. It was, I mean, back in the '80s. So this will be a much updated, better, newer and so forth. But anyway, we'll see how that goes.

Leo: Okay.

Steve: And finally, a nice note on a serious note from Philip Cooke, who wrote, "A week ago I started my day with a Blue Screen of Death" - not the way you want to start your week - "advising that I had an unmountable boot volume."

Leo: Oh, dear.

Steve: "Efforts by a Dell tech only led him to the conclusion that we should reformat the drive and lose all my data. Nothing would recognize the drive, and all the chkdsk commands in the book could not even see it or result in anything but the same blue screen on every reboot, no matter how I started. A Maxtor utility that I ran from a downloaded file advised me to return the drive for a replacement. After getting estimates ranging from $400 to $2,700 to recover my data, and trying numerous other 'tricks,'" he

has in quotes, "recommended by online chats, et cetera, I was fortunate to come across SpinRite.

"At first the glowing testimonials seemed just too good to be true. And I will admit that I thought they may have even been fake." Of course he doesn't know me very well, but that's fair enough. He said, "So I invested the $89 and downloaded the file and fired it up. At first I thought that it was going nowhere 'cause after four hours it still said 2% complete. I figured I would leave it running over the weekend, and imagine my surprise when I came in this morning, saw the message that it had completed. It booted up, ran chkdsk, and then started Windows. All I can say is wow. Thank you for taking the time to create this program. It's bad enough losing data, but I also saved the hours it would have taken to recreate my desktop, links, et cetera. Needless to say, I am impressed. Philip Cooke."

Leo: And image that drive right now, Philip.

Steve: Thank you, Philip.

Leo: Immediately. The chkdsk is superfluous after you run SpinRite; right? It's just doing - it's kind of a poor man's SpinRite.

Steve: Yeah, well, it's something that Windows often kicks you into when it's thinking, wow, we just recovered from something. Let's see if everything's okay. So it just is a perfunctory...

Leo: But if you've passed this, yeah, if you've passed the SpinRite test, there's nothing additional needed. But Windows will make you do it, of course. All right. Time to talk hash tables. Distributed, not just hash tables, distributed hash tables.

Steve: Yeah. And again, I'm unhappy with the label that this has received because it sounds uninteresting. Distributed hash tables is like, well, okay, I mean, and it's not even accurate because the hash tables are not distributed. The hashing is the way you access or you index a distributed database. So you'll see they got called distributed hash tables, we're calling them that because that's what you'll hear in the future. This is a technology which is part of the Internet culture now. We referred to it when we were talking about Tor. As we know, BitTorrent uses it in order to create a database in the cloud. It's the way that peer-to-peer networks and also networks that aren't concerned about a central authority shutting them down, which is one of the reasons people switch to a peer-to-peer model, but where, for example, in Amazon's case, they have something called "Dynamo." Dynamo is their own proprietary database technology based on distributed hash table technology.

Okay. So what does a hash have to do with a database? What we know about hashes is that you can put anything into a hash function, and what you get out is a fixed-length pseudorandom pattern of bits, is probably the best way to characterize it. We have talked, for example, in some cloud storage instances of, for example, hashing the contents of a file to create a key which would be then used to encrypt it. And what's clever about that is then the file provides its own key, and only somebody else who has the identical file could hash that to get the identical key. And then that's the way, for

example, a shared hosting, a shared cloud storage system could receive files that are encrypted that it has no knowledge how to decrypt. Anyway, so that's an example.

But the other thing you can do is you could take the name of an object and hash it and get a key, essentially. And so that's what distributed hash tables do is they use a hash as - it's also sometimes called a "mixing function." They just sort of mix the thing up into this random value.

Now, in the case of BitTorrent, you're wanting to identify files in a big database that various BitTorrent clients have. In Amazon's case, they're wanting to - they've got a massive website with incredible number of images and blobs of text and user reviews and, I mean, just look at how complex a page on Amazon is. Those are not pages that are created by some individual. They're all dynamically generated by software. And they're all pulled together from individual bits.

So Amazon uses their distributed hash table-based database in order to hold all of this. And I've read their whitepaper on Dynamo front to back. And their biggest concern is latency. They absolutely know that they have to present a page fast. Period. No matter what. And they've really gone out of their way to keep the latency of the delivery of their pages down. Other applications of distributed hash tables may have different goals. They may want much more redundancy.

So, for example, in a BitTorrent model, you have highly variable availability of this collective database as individuals on a whim come and go. In Amazon's case, they have got fixed servers, and they want to be able to take a server down and have all of its data become unavailable, but have the whole system continue to function. And if a server fails, even if it's a nonscheduled downing of the server, they want to be able to, similarly, have there be no effect. But in general, because it's them in a datacenter that's cooled, and it's got power and everything, they don't expect unexpected events at the same rate as, for example, BitTorrent's individual nodes coming and going from the collective 'Net.

So the common concept here is that you use whatever tag it is to identify an object. It might be one of those wacky-looking GUID, you know, Globally Unique ID, GUID strings. It might just be the natural name of the file. Whatever it is, you put it through a hash function in order to get the result. Now, Amazon uses MD5 because for them it's good enough. Now, notice, this is not an instance where the security of MD5 is a problem. This is somewhere where actually the speed of MD5 is a benefit. And the fact that it gives you 128 bits output is sufficient. If for some reason you had an application where you needed more than that, SHA-1 gives you 160 bits, and SHA-256 gives you 256 bits.

So the goal, regardless of what hash you use, is you're going to get out this string of bits. And the idea is that, and this is the key, and we understand listeners of the podcast will get it, that the point is, every time you put the same string in, you get the same pseudorandom pattern of bits out. So that's the key. The hash table, or the hash, it maps the input string into a pseudorandom output. And it does it fairly. And fairness is the key. That is, that the nature of a cryptographic hash is that, if just one bit of the input changes, if in the name you changed this from redpixel, you changed it to redpixels, or you made a tiny change, every bit that you change on average inverts 50 percent of the output hash's bits. So it is extremely sensitive to modification.

And the idea is that the hashing function, which is good, evenly distributes the results over the whole space. If you were using, for example, MD5, like Amazon has, 128 bits is 340 billion billion billion billion combinations. And the point is, obviously you're never going to have 340 billion billion billion billion servers to choose among. But you're going to divide this large number somehow into the actual nodes in the network which contain

pieces of your database. And we'll talk about that in a second.

But the idea is that the hash function doesn't discriminate. It's not going to choose some values over others. And that's what makes it a good cryptographic hash and not something that someone whipped up in their basement thinking, oh, this is a really good function. These are high-quality functions that guarantee an even distribution of results. So how do we turn this long bit string of ones and zeroes into a selector of a node. It's interesting, and this sort of harkens back to the oddness of the way the Internet works, where we've talked about how the genius of the Internet is that packets don't necessarily always get where they're going the first time, but the Internet layers redundancy on top of that in order to solve the problem.

Similarly, the designers of the distributed hashing index approach said, well, okay, if nodes are coming and going, then we don't know how many we're going to have at any given time. So how do we divide the total key space, that is, the number of possible key outputs from the hash, into the number of nodes? Because that's going to change. And also, if we use the key to index, to point to a node, then if we add nodes, how do we, like, rebalance everything? I mean, these are difficult problems when it comes to going from sort of the overall glossy, oh, look, let's use a hash, to fairly distribute storage responsibility across a number of nodes.

If you knew, for example, that you always had 16 nodes, say you had 16 nodes, but you just use the low 4 bits of the hash, it's that easy because 4 bits can have any of 16 values, zero through 15. And so you use the 4 bits, the low 4 bits of the hash, to choose the node. And problem solved. Except what if there's 17 nodes? Now you've got a problem because you were using 4 bits, and that only gives you access to 16 nodes. But you've got another node, you'd like to use it somehow.

So what they decided is sort of bizarre. The designers of these systems divide the total key space, the total space of the hash, which, for example, in MD5 is 128 bits, they may just choose, okay, say 32 bits out of that 128. We know that 32 bits gives us 4.3 billion combinations. So they then segment that into sort of ranges, some arbitrary number of ranges. Again, way more than they actually have nodes. And then they randomly - and this is the part that's a little, you know, creates some cognitive dissonance - they randomly assign these ranges to nodes so that, when a key is put through the hash, it results in a value that randomly - that chooses a node where the data will be stored. But that node has been chosen in a random fashion.

The way this happens is, imagine a clock where the end connects to the beginning, like 1:00 o'clock, 2:00 o'clock, 3:00 o'clock, 4:00 o'clock, 5:00, 6:00, 7:00, 8:00, 9:00, 10:00, 11:00, 12:00, and 12:00 wraps around into 1:00. So we have a circle, or a ring. And the output values of the hash function essentially form a ring from all the bits are zero, like at 12:00 o'clock, and increasing value as we go around this clock face until we get to midnight, or back around to zero, which is all one bits. So you can see how it's possible to map the values coming out of the hash into locations in a circle.

So the nodes choose some number of locations at random on the circle. And the number of locations chosen is a function of sort of the, as I was mentioning before, BitTorrent would use a different sort of strategy than Amazon's Dynamo. But it might be a hundred. So just say a hundred. And then the number doesn't matter, either. So each node chooses a hundred locations at random around this circle, where the circle is indexed by the output of the hash.

And the way the logic works, because the spots on the circle that the nodes chose will almost certainly not correspond to keys that come out of the hash - because we're

dealing with 340 billion billion billion billion possible keys. So the idea is, when we want to look up which node contains the data, we run the name of the file we're wanting to look up through the hash and get out this 128-bit value which identifies a location on the circle. We then go clockwise until we find the first node that chose that spot, clockwise from where we started, spot on the circle. That's the node that we ask for the data. And that gives - so that's sort of the primary owner of the data. Redundancy comes from allowing some number of nodes in succeeding locations of that circle to carry copies. So if we ask that first node and don't get a response from it, then we move, again clockwise, to the next spot on the circle that a node chose. And it will have a copy of the data from the previous node. So what this has done, wacky as it is, it works, is this allows us...

Leo: That should be the motto of this show: "Wacky as it is, it works."

Steve: Wacky as it is, it works. So nodes appear. And they say, hi there. I want to participate in this wacky shared database. So I'm going to choose a hundred spots at random. Now, their responsibility is to begin carrying the data from the four upstream nodes; right? Because all of the four upstream nodes are going to be relying on that node as successive levels of backup.

So a node appears, and it says, okay, here I am. Oh, and there is a metadata communication in the network. There's actual a protocol called "gossip," which is how the nodes talk. They gossip among themselves, not in a big hurry, because there's so much resilience and redundance in the system that this is not important if the news of nodes coming and going doesn't propagate instantly. It's all kind of very relaxed. So a new node comes in and says, hi there. Here's the hundred spots I've chosen, everybody. So everybody update your knowledge of me. And I'm going to go start copying, replicating, data from the four nodes that are counterclockwise upstream of me.

And so this node goes and asks the other guys for their data. And again, it's not in a hurry. It's redundant. So it would like to get the data, and it will in time, and hopefully long enough to be of use. And so the idea, then, is that clients, wherever they may be, who want to access the data in this database, they ask any one of these nodes - very much like a Tor node. Tor nodes would be typical distributed database nodes. They ask any of these nodes for the data. That node that's been gossiping with the other nodes has a maybe up-to-date, maybe it's a little out of date, doesn't really matter, but it's got sort of the latest news in the gossip circle. It hashes the key, uses its own most recent copy of the gossip news to identify where on the ring this would be, and that tells it which node is primary storage for the data.

It then asks that node for the data, saying hey, here's a key. Do you have the actual data corresponding to this key? And, if so, send it to this guy. And if that node says nope, I don't have it, then we know that there are redundant copies. Oh, and if the node doesn't answer at all, if it just disappeared off the network, it's like, oh, okay. That begins to propagate the news through the gossip channels. Meanwhile, we know that there are other nodes that have it.

And so what ends up happening is, with this very sort of lazy, relaxed, gossipy protocol, the data survives as individual nodes in this network come and go over time. It's duplicated. It's redundant. Anybody can be asked for any data, and they more or less, enough, know how to find it. And it's, again, it's crazy. It's wacky. But it works.

Leo: Node roulette.

Steve: That's how, yes, that's how distributed hash tables and the databases behind them function. It's a little unsettling for someone who likes to absolutely know where something is located. But the advantage is chunks of this can just disappear, and it doesn't matter. The whole thing continues. There's no single point of failure. And it can be tuned for speed. It can be tuned for resilience. It could be tuned for redundancy, whatever characteristics your particular application has. And if you look at the Wikipedia article on distributed hash tables, there's a bunch of projects. Apache has a project that is doing this, and Freenet, I think it is, is based on this. And so it is a technology of the Internet. I imagine we'll be talking about it in the future. And of course Bitcoin is the same thing. Bitcoin is a distributed network of individual miners, yeah.

Leo: Yeah. You don't have to be a miner to love Bitcoin. And now that we can accept bitcoin donations, I love it, too.

Steve: [Laughing]

Leo: Yeah. So, Steve Gibson, there you go, distributed hash tables, it's all clear as mud. And I hope everybody understood that. And if you didn't, well, you know what? You can listen to it again. You could even read it because we've got text transcriptions of the show and 16Kb audio available at Steve's site, GRC.com. We have on-demand audio and video, higher quality of both, available at our site, TWiT.tv/sn.

And if you go to GRC.com, do check out SpinRite, world's best hard drive maintenance and recovery utility. Check out the new Fingerprints feature. Check out ShieldsUP!, make sure your router is safe, all of that stuff. Everything's free except SpinRite. That's his bread and butter. So support Steve because he does such a great job every week on the show. We really are grateful to you, Steve. And if you go to GRC.com/feedback, you can ask questions. Steve will be doing a Q&A episode, security willing, next week. Hackers willing, next week. Thanks, Steve.

Steve: Thank you, Leo.

Leo: A lot of fun. We'll see you next time on Security Now!.