GoogleUpdate & DNS Security

Description: Steve and Leo wrap up the loose ends from last week's final Q&A question regarding the self-removal of the GoogleUpdate system following the removal of Google's Chrome web browser, then discuss the operation and politics of upgrading the Internet's entire DNS system to fully secure operation.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-163.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-163-lq.mp3

INTRO: Netcasts you love, from people you trust. This is TWiT.

Leo Laporte: Bandwidth for Security Now! is provided by AOL Radio at AOL.com/podcasting.

This is Security Now! with Steve Gibson, Episode 163 for September 25, 2008: DNS Security. This show is brought to you by listeners like you and your contributions. We couldn't do it without you. Thanks so much.

It's time for Security Now!, a chance to talk about the world of security as it applies to technology. Steve Gibson is here. He's the guy at GRC.com, Gibson Research Corporation. They do...

Steve Gibson: It's really not our listeners' chance to talk about it. It's...

Leo: It's our chance to talk to you.

Steve: ...our chance to talk about it, and their chance to listen. So, yeah.

Leo: There he is, Steve Gibson, always precise. You can tell he's an engineer.

Steve: Leo, I listen to you. That's the thing.

Leo: Now, that's true, too. Nobody else listens to me.

Steve: I pay attention.

Leo: But, yeah, that's actually your chief virtue and charm is that, like most engineers, you're very precise about your choice of words, and you try very hard to accurately represent the facts.

Steve: When you program in Assembly language, you have no choice.

Leo: It's that or nothing.

Steve: No room, comma, dot dot, semicolon, so forth.

Leo: So a lot to talk about today. We do have tech security news. We have some corrections to make. We're going to talk about Chrome because you've done now your analysis of Chrome. We also - we're going to talk about DNS security, you said.

Steve: Yeah. I had promised to talk about it weeks ago when we were talking about the whole DNS spoofing problem. And so I did some sort of catching up. And I discovered that there's more interesting things happening on the political side than really on the technical side. So I want to sort of talk about sort of the notion of the technology a little bit about DNS security, but what it really means to have secure DNS because it enables all kinds of other stuff that would be really good to have. But there's also some problems because turns out only one person can own it. And we've got a big, complex globe.

Leo: Yes. And...

Steve: You can imagine that would be a problem.

Leo: ...it's not just the U.S. that gets to decide this stuff, either.

Steve: Well, but as I'll explain, there can only be one owner. And no one really is happy with the U.S. being that person.

Leo: So what's in the news? I notice I just got a Windows update all of a sudden. Is this the second Tuesday of the month? No.

Steve: No. Interesting. I didn't check mine, either, nor have I seen anything written up recently. That's interesting, too, because there have been some complaints recently raised about Apple's nonsynchronous pushing out of updates. Apple is releasing updates

whenever they feel like it.

Leo: Well, so did Microsoft for the longest time, until there were so many updates they couldn't do it that way anymore.

Steve: Well, actually it was the corporate guys who said, look...

Leo: You're killing us here.

Steve: We can't have these kinds of interruptions on an ad hoc basis. We just can't have them appearing at the door and needing to stop everything and figure out what they mean and how important they are. So it was really, you know, end users don't care. But it was the corporate IT guys...

Leo: Oh, they care a little bit if they happen every day. But hackers don't wait, either. I mean, that's the problem. They don't wait till every 30 days to release attacks.

Steve: It is true that, to the degree that Apple is pushing them out the moment they're ready, as we know, that reduces the size of the window of opportunity for exploits. So on the pro asynchronous update-pushing side, that is, the Apple approach, you could say, well, yes, but it's minimizing the opportunity for exploitation. On the con side, though, now we're seeing, as Apple is trying to get themselves more into a corporate environment than before, and as Apple's market share is increasing, this is becoming a problem. And corporate IT people are pushing back on Apple, saying, look, look what Microsoft does. We like that. We'd like it if you did that, too. So, but we have been seeing Microsoft falling out of this second Tuesday of the month routine. There have been, like, little patchy patches coming out where they've made a mistake, apparently, with their initial update and then made a fix and then put it out again. So I've been seeing the same sort of thing you have, Leo.

I wanted to say at the top of the show that we've been getting a bunch of people trying to submit stuff to me, to Security Now!, who have been sending email to GRC. Well, that works, but we've got a place for it to go. And I realize, as I've been seeing this swell of people saying how do I send something to you guys - and in fact Leo you have forwarded things to me that people have tried to send - that we haven't been saying, I haven't been saying "GRC.com/feedback." So I wanted to make sure that everyone knew to submit things to Security Now! that sort of go into that bin, GRC.com/feedback. And that stuff, it's a web form that you can drop your text into if you've written it ahead of time. It's as anonymous as you want it to be. I don't, you know, I like to have names and locations so that we can sort of give a sense when you're reading the questions on our Q&A episodes where people are, who and where, it sort of makes it more personal and fun. These are real people, not disembodied text streams. But I just wanted to make sure everyone knew GRC.com/feedback was the way to submit things to us.

Last week we, in one of our - in last week's Q&A I read a question from someone that caused me to sort of raise some alarm about the uninstall behavior of Google's Chrome browser. The first time - and it's funny, too, because I even mentioned how skeptical I was of this initially. But when several people talked about it being the case, I then took a

look at it myself. And the issue was that upon removing the Chrome browser, there was stuff apparently deliberately left behind, namely all of the GoogleUpdate infrastructure which Chrome brings along with it, which is used for maintaining the currency of any of Google's code base, whether it's their desktop search or their web browser toolbar or the Chrome browser. And in this instance I had set up a clean virtual machine that had never seen any Google code at all, installed Chrome, removed Chrome, and sure enough, there was all this stuff left behind. There was essentially GoogleUpdate.exe was still invoking itself in the Run key of the registry, so that it would always arrange to be running all of the time. And that continued for hours after Chrome had been removed.

So I verified that behavior on the show last week, and I said I'd be looking into it more deeply, but this looked like a problem. Not long afterwards, that VMware session was still open on one of my monitors on the side, and I happened to glance over, and it was all gone. I mean, completely perfectly beautifully removed from the system. So it's like, okay, wait a minute. Did I push something, or did I do something? So I immediately retraced my steps, recreated the experiment, and sure enough, removing Chrome, this stuff was left there. I restarted the system, the virtual machine, a few times. I watched with a packet capture, and I saw Chrome phoning home to the mothership and exchanging, through an XML dialogue, exchanging a bunch of gobbledy-gook...

Leo: You can't tell what it's saying, can you. I mean, is it - it's not obvious what they're saying. You can't say, oh...

Steve: Right, it's not English. Not any English that we know. And so finally, so I immediately sent email to you guys there saying, whoa, we need to head off the alarm bells from last week because I don't understand exactly what's going on yet, but it looks like it's removing itself, for some mysterious reason, maybe on some criteria based on its conversation with Google, or I don't know what. So the most...

Leo: That's the interesting thing. You can't tell. It's like not a certain number of reboots or something. It's just...

Steve: Yes, it's not. And then I thought, okay, as I was about to say, my most telling experiment was that I installed Chrome, and then I shut down that virtual machine's connection to the Internet so that neither Chrome nor GoogleUpdate could phone home in any way.

Leo: Okay.

Steve: Then I removed Chrome. And instead of getting the popup screen, oh, ow, please tell us why you are leaving us, I just got a "could not display this web page." And I said, okay, good. And now neither could GoogleUpdate phone home. I just wanted to see whether it was disappearing on a schedule, if it was disappearing when it had permission to leave, that is; and, specifically, if it had been abandoned, if it was left behind on your machine, was there any scenario that would keep it from removing itself. Well, for one thing, if you kept it from running, then it wouldn't remove itself, although it has many ways of running because it does install plug-ins into all the browsers it can find to keep its hooks into your system. But if you just do the normal thing, that is, you uninstall Chrome and, literally, wait a while, even with no connection to the Internet, it tries to

phone home; it can't resolve the domain name that Google is using for that purpose. So after a few hours it goes away. It just cleans itself up and says, okay, I guess there's nobody here who needs me or wants me anymore, so I'm leaving. And it does. Now, I've got to say I have no problem with that. It'd be a little nicer if it noticed that it was the last piece of Google code in the system and went away. I don't know why it couldn't do that. But I would imagine maybe the Google guys have some reason, or maybe they're thinking you're going to reinstall afterwards, they don't want to...

Leo: Yeah, they want to give you a chance, yeah.

Steve: They want to give you a chance to change your mind. I don't know what they're thinking or why. Or maybe they want some final chance to do some final, after-removal reporting back to...

Leo: Yeah, there may be some reboots, too, that need to go on. I mean, it's odd that it wouldn't be - I could see how it would be after one reboot it might go away.

Steve: I tried once, multiple times, none. It just seems to be based on time. And not even seeming - I didn't even - I didn't watch the time. Because normally I would look up, and it had taken some opportunity to disappear between then and the time I had looked before. So it's hard to keep track of it. Who knows. But the point is, it is not the case, and this answers the question from last week, not the case that Google leaves stuff forever in your system.

Leo: And we should reiterate, and I said this - I should explain what happened. So those of you who saw it live saw us go on and on and on about this. And then, before we put out the podcast, Steve emailed me back with saying, wait a minute, hold on, it did seem to uninstall itself. Let's do some research and come back next week. So I essentially cut it out. I didn't want to scare people without having the facts in front of us. So I cut out most of that discussion. And I did say this, but I want to reiterate this, that the Chrome folder that our correspondent saw, and this is confusing, is not Google's Chrome, but it's what Firefox calls its UI. So it's a bunch of JAR files for the Firefox UI. So, yes, deleting that Chrome folder does break Firefox. But you do stand by the points that Chrome does install something into - a browser helper object into Internet Explorer. It does install a plug-in into Firefox.

Steve: Right.

Leo: And it does install the GoogleUpdater. It does phone home with stuff we don't know what it's saying. The only thing we correct is - and I think that, because I cut that part out, I don't think it's a major correction. But the only thing that we correct is that the updater does uninstall itself after time. It leaves itself on there initially. But after some unknown amount of time it uninstalls itself.

Steve: Yes. And I did - and maybe this was also something that was removed from the podcast. I did listen to what you had edited, Leo, and it sounded much better to me. Of course it sounded like...

Leo: And in the case of - I absolutely want complete transparency here. So we talked about it after the video so that people understood who had watched the video. I hope people stayed tuned for it. And then I just, you know, in this case I feel like it's as if a newspaper had an article, something in the article didn't fact-check out, took it out before publishing it. And I feel like that's what we did. But because we broadcast live, people saw the sausage being made. So I just want to make it very clear exactly what happened so that you guys understand. And I just didn't feel like we should publish it with something that we didn't understand yet. And now we're coming back and explaining it.

Steve: Oh, absolutely. And the fact that we have the listener base of the podcast is substantially larger than the viewer base.

Leo: Absolutely, yeah.

Steve: Yeah. We did also mention, or there was some - I agreed certainly in talking to you that I would take a look at Apple, too, because there were some people in the chatroom during the podcast that were upset, saying, well, I mean, Apple does the same thing.

Leo: Which is no excuse, obviously. It's...

Steve: Exactly. I mean, I feel that very strongly. So I don't care what Apple does.

Leo: We'll call them on that, as well. That doesn't make it okay.

Steve: I did take a look, though, through a Safari install and remove. And Apple operates differently. I actually like the way Apple's update functions more than Google's. Apple Update functions by registering a task scheduler job to run, in my case, late Saturday night once a week. So only - so nothing is running in the system all the time, which I like a lot more than Google Update sitting there as a process. I mean, I try to avoid...

Leo: Too many programs do this, and it slows your system down eventually because you have all these processes sitting there.

Steve: Yes, and essentially everyone seems to think that they own your system and have a right to come in and stomp around on it, which is really annoying for someone who is trying - for any users who are security conscious and want to have some sense for what all this gobbledy-gook is that's running in their system. It slows your boot. It's burning up some RAM resources.

Leo: And it impacts reliability. And it makes it very difficult to detect spyware

because you see all these processes running, and you don't know what belongs to what.

Steve: Yes. To GoogleUpdate's credit, I will say that it is not big. If it were hundreds, well, not hundreds of megabytes - well, yeah, that, too. But, I mean, if it were really big and running all the time, that would be a problem. But in my case it was about 183K. Which is...

Leo: That's tiny.

Steve: It's, yeah, compared to the amount of RAM these days it's tiny.

Leo: It generally seems to be the case on OS X, and I think it's true on Linux, that programs - the kind of default way of doing this is that Chrome would check on start. It wouldn't keep something running all the time. But when the application runs, it would check for updates and then download updates. And most Apple applications work that way. They download updates only when they're run. They don't stick stuff in the background. And I think that's true on Linux, as well. It's just kind of not done.

Steve: Well, and so we've heard, and this may just be anecdotal or initial industry stuff, I remember hearing that Google was looking often to, like, GoogleUpdate was checking hourly to see if there was anything new.

Leo: Now, in their defense, what they're checking for, they say, is new phishing pages, new secure - they're looking to update their phishing thing. And I think Internet Explorer 7 does something similar; doesn't it?

Steve: I have not looked closely.

Leo: I think it does update its database of malware sites. And god knows there's new ones all the time. So maybe you could make the justification. Seems to me, though, it should be doing that when it's running. I mean, if you only run Chrome once a month to run Gmail or something, why do you want something running in the background every hour, updating Chrome?

Steve: Right. And again, this can also be, in my opinion, covered by the fact that they're in beta mode, that this is a beta. Lord knows how many years it's going to be labeled "beta," given Google's reputation or penchant for leaving things in beta forever. But anyway, I did want to mention that what Apple does is it installs the Apple Update, which it does not remove when you remove Safari. It does, however leave it in your Add/Remove Programs listing. So if you install Safari and remove Safari, the act of installing Safari does bring Apple Update along for the ride, but does not take it out. Now, again, I don't know if next Saturday night, when Apple Update runs, if it would notice that Safari had left, and then it would remove itself. I didn't check that. But I did

see that there were two separate entries in Windows Add/Remove Programs files listing, one for Safari and a separate one for Apple Update. And you just click on that and remove it, and it goes away. So they're operating very differently. I like the idea that Apple Update's independent updater is not running all the time, and it runs weekly, which seems fine to me. And as you said, Leo, certainly when you run Safari, Safari can check to see if there's anything...

Leo: That's a good time to do it. Maybe they do it because they want to speed up boot, you know, browsers, you want them - look how fast Chrome pops up and is ready to go. And that's probably why they don't want to do that on startup. But that's the price you pay is that your machine is always doing it every hour, whether you want it to or not. So on the balance, now, knowing what you know about Chrome, are you still concerned about its security and about whether people should install it?

Steve: Oh, yeah. I'm not at all concerned about the GoogleUpdate behavior. I think that's entirely acceptable, especially as a beta. If it had left the stuff in the system forever, then even as a beta I would feel that's unacceptable. But this is just fine the way it is now. However, I still feel the same way about the browser being too feature-lean. It's unacceptable for it to, by default, to offer to save website passwords and provide no facility for the owner of the system to protect those from prying eyes.

Leo: That's a real problem.

Steve: I mean, that's just nuts. And, frankly, I know that Google wants this to be a platform for running JavaScript applications and sort of the Web 2.0 model of things, the browser as the application platform. But it is, I mean, all the vulnerabilities we run across today are scripting driven. They are enabled by scripting, by going to malicious sites that use scripting in order to cause something bad to happen in your system. Okay, I said "all." I shouldn't have said "all" because there, for example, we just had image-based vulnerabilities with GDI+ in Microsoft's last update. So there was a non-scripting vulnerability where just showing an image would cause, could cause a remote code execution exploit. So it's not the case that absolutely all, but 99.9 percent of what we've talked about in the last year have been script-driven exploits. So the idea that there's no way to disable JavaScript at all, let alone site-specific disabling, which IE offers natively and which the NoScript plug-in to Firefox allows, just the idea that it doesn't have it, it just - it can't compete in the browser world. And some people have said, well, it's not really a browser, it's an application platform. Well, yes, it's a browser. I mean, of course it's a browser.

Leo: When it comes to security, if you're using it on the Internet, I mean, I guess if you stuck with Google applications only, these would be less important. Of course it'd have your password, your Gmail password visible. I don't know if you really want that.

Steve: Yeah, that's true. And you talked about wanting to keep it around as a better platform for running Google Mail, for running Gmail, which I think makes absolute sense. I just think it's not ready yet to be someone's only browser.

Leo: No, no, no, no.

Steve: And my real big news, well, for me, I've switched to Firefox, Leo.

Leo: What? And why, Mr. Steve Gibson? Because I've had this battle with you for now nigh on four years.

Steve: As long as we've known each other, actually.

Leo: You've always been an IE user. Now, to your credit, I've always used you as an example of how you can use IE safely. And you do that. But why did you switch?

Steve: I just think that Firefox has become mature enough. And, frankly, there is a fabulous ecosystem of plug-ins that surround it. So I've spent some time in the last week - I just made the decision. I like the way it looked with Firefox 3. I liked the security improvements that they went - in going from 2 to 3. Firefox is one of the only browsers, I can't remember if it's the only - we'll be talking about privacy-related issues of browsers before long. But it's at the top of the pile of privacy enforcement. It really does a good job.

Leo: Well, except that its third-party - do you think its third-party cookie support is now acceptable, or blocking?

Steve: I wish it were - I wish third-party cookies were disabled by default. But they're not. That's my only complaint because it does block - it blocks outbound cookies, which is outbound third-party cookies, which you want, not blocking inbound third-party cookies, which unfortunately the WebKit-derived browsers, specifically Safari and Chrome - oh, and IE, which is not WebKit derived, but IE also - they block incoming but not outgoing, which is wrong, the wrong approach to blocking and filtering cookies. So but more than that, if you add a couple simple plug-ins like NoScript - and people have been jumping up and down about it, I mean, I've been hearing so much about it, I thought, okay, I've got to find out what this is. Time to give up on IE, Gibson, and switch. And I have switched.

And finally it took - after a couple days I realized when I was clicking on links I was still getting an IE instance. And finally I said, okay, time to switch over to Firefox full-time. I mean, so Windows Update will still run IE in order - because, you know, it will only run in IE, which is just fine. I'm doing all my surfing on Firefox 3, and I'm very happy. And using NoScript with scripting disabled by default, and then you selectively enable it for those sites that need it, I mean, that's the way to surf.

Leo: And I should point out that Steve is the opposite of an early adopter. But that's - I think anybody who is really concerned about security, that's the way it should be. I mean, you only recently moved off of Windows 2000 to Windows XP.

Steve: Yeah.

Leo: And Vista, well, Vista is way off in the future. Vista is maybe never.

Steve: I wanted to mention, since we had talked about the large Hadron Collider recently...

Leo: Oh, bad news there, huh.

Steve: It's shut down. Yeah, there was a - I loved it, too, because what it literally had was a coolant leak. And I thought, this is the Enterprise. Because we were talking about how it's like a starship when you look at the photos of this thing. And of course they were always having coolant leaks. That's bad on a starship. But it turns out it's just as bad on a superconducting magnet. When you get a coolant leak on a superconducting magnet, all kinds of things melt; and you get helium, I think it's helium that was leaking into the internal chambers. Anyway, they are now shut down until spring because the electricity costs so much, and the rate, the cost of electricity goes up in the winter, that they're now shut down until next spring, when they are able to afford, well, they have to make repairs and then afford to fire this puppy back up again. So not...

Leo: [Sighing].

Steve: I did also want to mention in my little own errata section that I was really delighted when I happened to go back to the Kindle's online shopping area, looking at newspapers, and the Financial Times is now available.

Leo: You read the Financial Times?

Steve: It is, in my opinion, the best of the financial-oriented newspapers. I've been reading The Wall Street Journal a lot. And then I added The New York Times to - I already had the L.A. Times I was looking at. But I even have my own little county's - the Orange County Register, of all things, is available now. There's a much-expanded collection of newspapers. And I have to say, for any of our listeners who know the Financial Times, I've been now comparing them side by side during this last week of financial roller coaster.

Leo: Oh, I bet that's an interesting read. Because it's British. It's out of England. Isn't it?

Steve: I don't know. I wouldn't be surprised. I've sort of sensed some of that in like the way they use the pronouns and things.

Leo: Little bit proper.

Steve: But what I like about it is that it feels like the writers really understand what they're writing about. Whereas even The Wall Street Journal has a little bit of feeling like, well, we're sort of rehashing what came over the AP newswire, and we don't really understand it. And what I find is, as a consequence, I understand it better when I read it from the Financial Times than I do The Wall Street Journal. Which I was assuming was the gospel, essentially.

Leo: I'm going to defend the Journal because I have some friends who work there. I think that they do actually know - they have some of the best reporters in the country. It's a very good newspaper. I don't read the Opinion page because my politics differ. But it may be that, you're right, that maybe they aren't covering - I wonder, you know, because they're in the back pocket, aren't they, of Wall Street? So I wonder if their coverage is perhaps a little less objective than something like the Financial Times.

Steve: It might be that. I'm also seeing that the articles, the stories in the Financial Times are shorter. The Kindle gives you the word count at the top of each story. And, for example, this morning I read everything that I needed to during my morning Starbucks coffee sojourn in the Financial Times. And then I quickly scanned the front pages of The Wall Street Journal, The New York Times, and the L.A. Times. And there were some interesting local stories. We finally got our budget signed by the Governator yesterday, which is a good thing for the California budget. And but nothing really else that I wished I had read elsewhere. So the fact that the Financial Times is being covered in the Kindle or is now available in the Kindle I think is just really good news that I wanted to bring to the attention of our listeners who, you know, the four of them who might care.

Leo: Hey, I lent my Kindle to Megan. She wanted to do a review. And after two weeks I was on my knees, saying, can you bring my Kindle back? I miss it.

Steve: Yeah.

Leo: I now do not, you know, in fact I bought a paperback book and started to read it. And then I said, wait a minute, I bet this is available on Amazon's Kindle. And I did, and I ordered it, and I put the book aside. It's just easier to read it on the Kindle.

Steve: I know, it's so nice, well, and so universal, too, to carry it around. And frankly, I mean, books are great to read. I've read many on it. But for me, you really get the leverage of the modem, and the articles are just there. When you're subscribing to newspapers, and they're just there in the morning, it's just so nice. And in fact I was mentioning to somebody, oh, it was somebody at Starbucks, about the Financial Times. And he was complaining that it didn't arrive physically until noon for him.

Leo: What, at 3:00 a.m. you get it on...

Steve: Yeah, it's just in the Kindle in the morning. It's certainly there by 4:40 a.m. when I'm up and rolling, so...

Leo: I love it.

Steve: And in other interesting security-related news, Sarah Palin's Yahoo! email account...

Leo: She had, what, gov@ - or alaskagov@yahoo.com? I can't remember...

Steve: She had gov.palin and gov.sarah, those two accounts both at Yahoo.com. And of course it was in the news last week that her account was somehow broken into. And it turns out what this person did, and we're thinking unfortunately it's the son of a representative, it's pretty much...

Leo: Oh, you're kidding.

Steve: Yeah, the FBI tracked him down. He used a proxy in order to get some sort of IP coverage, apparently.

Leo: But he didn't use TOR. He used a one-step proxy.

Steve: Yes, he did. And it turns out that the IP that he bounced through is handled by the ISP that covers the school that he attends.

Leo: Oh, he's a kid.

Steve: Yeah, he is. And anyway...

Leo: Just shows you how easy it was because it was social - it was basically social engineering, wasn't it.

Steve: Well, yes. And the reason this is a really great topic for us, I mean, we're not going to spend the whole time talking about it, of course, but it brings to light an interesting problem. Essentially he went to Yahoo! and figured out what the account name was, that gov.sarah or gov.palin, or maybe he knew from somewhere, I'm not sure how that was determined. But there of course was a password that he needed. So he told Yahoo! that he had forgotten the password and so went through Yahoo!'s password recovery steps, which involves answering a bunch of so-called "secret questions." The problem is, these were not very hard questions for someone to know the answers to. One was...

Leo: If you've been following the story, you know a lot of them.

Steve: Well, and in fact you don't even need to follow the story. I mean, if you arrive late to the party and just Google her or Wikipedia her or do basically any publicly available source of information. So I think one of the questions was where did you meet your husband, and so that was something that she had chosen probably from a list of standard questions. And this kid guessed from whatever information he had that she met him in high school. And he knew where she came from, so he knew what town that was. He put that in. One was what is your zip code, and one is your date of birth, both which were available publicly because she's now a celebrity.

Leo: Right, right.

Steve: And it said, yeah, okay, fine. What would you like to set your password to? And he set it to "popcorn." And so...

Leo: Now, this same kind of thing happened to Paris Hilton with her T-Mobile Sidekick. And it's because, if you're public figure - and the secret security question was what's the name of your dog, which she talks about all the time. So if you're a public figure, this information is known. You can't use the obvious answers to your security questions.

Steve: Right.

Leo: I mean, do you think it's Yahoo!'s fault? They should have a better system?

Steve: It's everybody's fault. I mean, you would argue, I mean, I have no position one way or the other about Sarah using Yahoo!. That's questionable, though, whether official business should be conducted in a public, nonsecure forum like that. There have been partisan people who have been criticizing her, assuming that she was doing this in order to avoid whatever records maintenance is normally - someone in the government is normally subjected to for the sake of maintaining public records for posterity. I don't know about any of that, and I have no opinion one way or the other. But what's of interest to our listeners, I think, is this idea that the password recovery questions could essentially be so insecure. So is Yahoo! responsible? I don't know. But certainly we've seen situations where you do password recovery choosing from a list of questions. There are other ones where you are able...

Leo: My bank does that. I mean...

Steve: Right. There are others where you're able to provide your own question. And I think that's probably more useful. You can ask yourself a question that only you would know, as opposed to here's a bunch of suggested questions, none of which tend to be very difficult to answer.

Leo: Well, you don't have to use the right answer, either. You know, it asks me frequently what was my first pet. I don't use - first of all, I've never told anybody the

name of my first pet. But in case I do, I'm a public figure, I talk a lot. In case I do, I don't use that as the name. I use some other name. So but then you have to remember it. It gets complicated. So...

Steve: Security always comes at a price. The last thing I wanted to mention was actually from the SANS newsletter. It's something I picked up on last week, but I just sort of moved it over for our conversation this week, I thought was very interesting. They reported three different instances of - and this is just in the last couple weeks - of employees of one sort or another being responsible for breaches, serious breaches of their own company security. And I wanted to share these with our listeners because this is something we have not yet, in all this time, spoken about.

The first one was titled "Former Intel Employee Charged with Theft of Trade Secrets." And it reads, from the SANS newsletter, "Former Intel Corp. employee Biswamohan Pani has been charged with theft of trade secrets for allegedly stealing proprietary company data, including information about the development of new chips." This is, you know, he's an Intel guy. Says, "Pani allegedly accessed an encrypted system at Intel and downloaded 13 top-secret documents. He had resigned from Intel in May and was taking accrued vacation time through June 11th. The intrusions occurred between June 8th and 10th." So in the last three days of his - nominally his employment prior to June 11th, when it was finalized. "Pani had already begun to work for Intel competitor AMD. The issue was discovered when an employee looked into Pani's access and download history on the system in question."

Leo: Oh, boy. Oh, boy.

Steve: So there was one. Then "Countrywide Notifying Customers of Data Breach. Personally identifiable information of as many as two million Countrywide customers may have been sold by data thieves, according to the mortgage company. While there have been no reports of the information being used to commit identity fraud, Countrywide is offering two years of credit monitoring to affected customers." Oh, thanks a lot. Guess that's better than nothing after the information's escaped. And of course the customers were notified. "The data were allegedly stolen by a former Countrywide employee who downloaded approximately 20,000 customer records every week for two years."

Leo: Oh, this is like embezzling, almost.

Steve: This is serious. Yeah, so this is, I mean, this is long term, 20,000 customer records every week for two years. Each batch of 20,000 customer records was allegedly sold for $500 U.S., or about..."

Leo: See, I'm a former Countrywide customer. So they would have my stuff; right?

Steve: Well, two million Countrywide customers were affected by this over the course of two years. And so this person was getting 2.5 cents for every record that they stole. And then it says, "It appears that the data were sold to other mortgage brokers."

Leo: Oh, that's interesting.

Steve: Yeah. So if you've been getting some unsolicited...

Leo: I get a lot of unsolicited - I get that all the time, I get unsolicited. But I thought that that was just because a loan is public record or something.

Steve: Didn't go off to Russia in this case, Leo, it just went to Countrywide's competitors.

Leo: You know, he's only getting 2.5 cents. It's what, like 500 bucks per download. I guess that adds up. But still.

Steve: 500 per batch, yeah.

Leo: Per batch, yeah. I guess that's...

Steve: Finally, the last story is "Insurance Office Employee Allegedly Used Customer Data to Open Accounts. Attorneys General in 45 U.S. states have been notified that a State Farm Insurance employee in Surprise, Arizona" - love that it happens to be in Surprise, Arizona, he was surprised - "used customer information to obtain credit cards. The compromised data included addresses, Social Security numbers, driver's license numbers, and in some cases bank account numbers. A company spokesman did not specify the number of people affected by the breach. Police are investigating. All affected customers have been contacted and offered one year of free credit monitoring."

Leo: Surprise. Surprise.

Steve: So this really speaks to an issue that, as I said, we've never directly addressed. and we certainly will at some point, and that is that we've talked about the notion of, I mean, all the security that we've really talked about has been how to protect from external intrusion, you know, people on VPNs, external passwords, the idea of given that everybody inside is doing the right thing, how do you protect your goodies from any bad people on the outside getting in. And these three stories I thought so perfectly demonstrate that there's a big problem, and in fact it has been argued this is the bigger problem, than dealing with bad guys on the outside who don't know your secrets. And that is dealing, you know, protecting your secrets from your own employees who have access.

Leo: I think that's almost a given that most jobs are inside jobs; right?

Steve: Yeah.

Leo: Yeah, I mean, I think that we've been saying - I've been saying that about hacking for a long time. You know, they're always looking to the outside guy. And pay a little more attention to the guy inside because that's who really has all the information. Things like letting this guy quit, go on accrued vacation time, and not canceling his account to the secret servers? That's just bone-headed.

Steve: And he already had a job at AMD. So, I mean, and we're of course completely speculating here, but he might have realized, wait a minute, I've got my accrued vacation time, technically I still have access because I'm really still an employee. And so who knows what pressure he was subjected to, either by AMD or his own ethics or lack of, I mean, it's impossible to guess that. But...

Leo: And we should probably mention that he's almost certainly broken the law and broken his employment contracts. And there will be some price to pay for this. I mean, it's not a - he's not going to walk away from this.

Steve: No, no, no. I mean, this is 13 top-secret documents to which he had access while an Intel employee.

Leo: I think that's a felony. I may be wrong, but I think that's a felony.

Steve: It's way bad.

Leo: Wire fraud.

Steve: Yeah.

Leo: Okay. So our topic of the day is DNS, and in particular how difficult it is to do DNS security. Before you do that, do you want to - do you have any SpinRite letters, anything else you want to talk about?

Steve: I did have one really nice, fun, happy story. And this...

Leo: From a Navy SEAL or a...

Steve: No, this is a Marty Sasaki. He wrote - he's in Arlington, looks like Arlington, Massachusetts. And he just said that - the subject was "Happy With SpinRite." He says, "I've been looking at SpinRite since I first heard about it on Tech TV," Leo.

Leo: Oh, my goodness. Oh, my goodness.

Steve: But, he said, "I couldn't justify spending the money on it until today. My system died. Can't tell whether it's a motherboard or a CPU problem. And the system is old enough that getting replacement parts is almost impossible. So instead I got a new motherboard, CPU, and RAM, as well as a new SATA disk drive, a Serial ATA disk." So, he says, "Time to copy over the PATA, the Parallel ATA disks. I kept having problems doing the copy, trying both PartitionMagic and GParted. It would always get a ways into the copy, then fail. So I figured I had a hard drive problem. I downloaded SpinRite and fired it up. It found a few errors, but it detected that one of the drives was running hot, really hot. SpinRite would stop, display the temperature warning. I would let it cool down a bit and then resume. Finally," he says, "I pulled the drive out of the case, mounted it so that I could blow a fan across it so I could get the data off of the drive. I probably would have figured it out eventually, but I saved myself hours, maybe days of playing around by getting SpinRite. I've got a couple of flaky disks around that I'll check out next. Perhaps they're still usable. Thanks."

Leo: I've had that happen, too, where the - what do you set the temperature for, default, for SpinRite to complain about the drive?

Steve: Boy, I don't remember now. When I set the spec for it, I did a survey of all the manufacturer specs and used the top end, that is, the high-end operating temperature that they were saying do not use a drive if it's hotter than this. So I'm not overreacting.

Leo: It's not a conservative number, by any means.

Steve: No, no, no, I mean, it's hot. And the idea being that I wanted to alert people, and this happens often, where they've got, for example, a case that was working fine. Then they said, oh, I need more terabytes of data. So they put higher performance, higher spinning, faster seeking, you know, more power-consuming drives in where they used to have a case that no longer has, as a consequence of that, adequate ventilation. So they end up with drives - and sadly, nothing else in the system, I mean, there are some add-ons you can get sometimes that will give you something in the tray running that will monitor your drive temperatures. You know, SpinRite does.

And the other thing is that oftentimes drives run hotter when they're being used. SpinRite uses the drive constantly. One of the things that does cause power consumption, and thus heat dissipation, is seeking. And so if the drive is cool when it's just sitting there doing nothing, and especially in the beginning when you power it up and the BIOS checks the drive's temperature, it says, oh, drive's cold. Yeah, how long is it going to stay cold is the question. So SpinRite does constant monitoring of the drive temperature in addition to all the other SMART data, the SMART, the Self-Monitoring Analysis and Reporting Technology stuff, while it's running, and alerts people if the drive is getting too hot when they're running SpinRite. And it's generally something you need to pay attention to, as this guy did. He finally arranged to run a fan over the drive to keep it cool enough in order to get SpinRite to fix the drive so that he could do his data transfer.

Leo: You know, it's funny, I could have written that letter. We had a computer that's been flaky for two years. And Colleen - I said, you know, we should really SpinRite this drive, and Colleen did. And it was overheating. And it's because it's in a Shuttle

case. And I think the Shuttle case is a little tight.

Steve: Yes, Shuttles actually have a big problem with adequate ventilation for the hard drive because it's underneath the floppy, and there is no real forced ventilation across the drive, you're right.

Leo: So she had it open. It was kind of funny. She had it open and a fan, like a table fan, blowing at the drive so that she could finish the SpinRite. But now we know we need to leave the case open and maybe have a fan blowing across the thing. We've actually retired that Shuttle. It was always flaky, and now I know why, it was too hot. So that's a useful little informative dialogue.

All right. Let's get back to the matter at hand, DNS security. We all became aware of DNS security when this big bug that Dan Kaminsky found showed up. He talked about it at Black Hat. People were stunned to find out that their Internet service provider wasn't providing secure DNS. It seems like we fixed the problem. Is it fixed?

Steve: Well, no.

Leo: Oh, great.

Steve: We've hugely mitigated the problem of DNS spoofing. But we haven't fixed it in the sense, for example, that we use SSL right now for really creating uneavesdroppable and authoritative connections to remote servers. As we know, SSL uses a certificate-based system, both to create an encrypted connection between two endpoints and to allow us to authenticate that we are connecting to a known remote server. So that, for example, PayPal gets a certificate which has been signed by someone whom we trust. And often VeriSign or Entrust or any of these certificate authorities, they go to hopefully significant lengths to verify that this is really PayPal that they're giving the certificate to. Then PayPal has the certificate on their server which our browsers connect to and verify that the certificate matches the domain name. Thus we get authentication of the endpoint that we're connecting to, as well as, thanks to the SSL protocol, we get encryption which protects it from being spoofed.

Now, the DNS system, the Domain Name System that we've talked about now several times, is a really flexible distributed database. It can contain much more than just DNS. And this is one of the really interesting ideas about how DNS could grow in the future if only it were more secure. Right now it's - you could argue, well, it's secure enough to look up IP addresses, especially where we add then an optional layer of SSL security to verify the identity of the remote endpoint to reduce the spoofing problem. But we know that there are still things that occur, like phishing attacks, for example, where users are misled to go to the wrong site. That's not a failure of DNS, although the problems that Dan found, the idea that DNS could be spoofed does make these problems more severe.

What DNSSEC, as it's called, which is the acronym for DNS security, what it offers is, for the first time ever, is cryptographically signed and authenticated DNS records, meaning that when a resolver, a DNS resolver is trying to get the IP for a domain, it could use a system very much like the system we have in place for security certificates. It could use

a system using public key technology, so-called PKI, the Public Key Infrastructure, to provably authenticate the contents of the DNS data that it receives so that it's able to absolutely state that this data is the same as was being offered by the authoritative name server.

Remember that the problem that Kaminsky exposed was that it was possible to do so-called "cache poisoning," meaning that the way DNS operates, it gets its scalability because you're not always having to ask the authoritative server. You're able to locally cache the information which you once obtain from the authoritative server, so you're not having to bother that authoritative server all the time. That's really where DNS is scalable, that is, the idea that you're able to cache this data. So what you want to verify is, if you get the data from a cache, is that the same as the authoritative name server has? And then you also have the problem that DNS uses no secure connections. It doesn't even use TCP connections. It uses UDP packets which, because it's just a query in one packet and a response coming back, they are infinitely spoofable. So even if you ask the authoritative name server, that is, not a cache, but the actual source of the DNS information for a given domain or zone, as it's called in DNS parlance, even if you ask that server, you'd have no way of knowing that you've got the response actually from the server because it's just a UDP transaction, a packet for a query and a packet for a response. So even there you would like to have the data that you received digitally signed so that you can absolutely verify that this is information you can trust.

Well, it turns out that essentially the same sort of hierarchical structure that DNS uses for its domain names, where for example you have www.paypal.com - and actually there's sort of a, even after the ".com" there is a root above the .com, where you have a hierarchy. You've got PayPal's name server that knows about PayPal and the PayPal domains. Then you've got a level above that is the .com servers that know about the .com domains, that is, all of the domains that are children of .com. And then above them are the root servers that know about all of the top-level domains, like .com, .org, .edu, .gov, and so forth, .mil, you know, all the domains underneath the root. So the system that was developed has evolved over, boy, I think we're now up to about 15 years. It's taken some time. And the early RFCs, the early documents that defined the first architecture, were found to be defective because they did not scale well.

One of the early concepts was that when a zone, meaning the records in a domain, changed, they needed to be resigned. The concept was that the parent would sign the children's records. So, for example, I'm at GRC. And if I changed some data in my GRC domain, I would have to have them resigned by the parent. And so the idea was that I would, in the original architecture, which has now been obsoleted, I would have to send those up to the, for example, the .com servers to sign. And then the .com servers would send signed records back. And the nature of the cryptographic signature was such that then I could provide those records to third parties, anybody who wanted the GRC records, and they would see that they had been signed by the .com servers, and they would be able to get the public key of the .com server in order to verify that the .com server had signed those using its private key.

Well, there were a number of problems with that. One is that that would require that the private key be available online, which was a huge concern because, if it was available online, meaning it was sort of like dynamically available, then there's a much greater chance that it could leak out, which would be catastrophic. But the biggest problem was that this would put a huge burden on all of the parent servers, any parents that had many children, since all of the children would constantly be needing to have their records signed. That was a huge problem. So a different approach was created where - essentially using some hashing technology, where instead of needing all of the individual records to be signed, it would be possible for a signature to be signed so that the domain

wanting itself authenticated would create a signature, and it would just send the signature up to the parent, that would sign the signature and send it back.

So there has been some evolution of this over time. But fundamentally what this would mean, if we could get DNSSEC working correctly, that is, get it broadly deployed and supported, it would mean that we would have, for the first time ever, a really, really secure, cryptographically securable distributed database for the Internet, which we've never had before. DNS has been okay, but with all kinds of problems, and not something you could trust. It's never been trustable.

Well, when trustability arrives, then we get a lot of things we've needed. For example, DNS, because of its flexibility, could be used for distributing public keys. There's never been a good means for distributing public keys. And that's, for example, been one of the things that has prevented email security from working well. If I wanted to use secure email on a broad basis, I have to have some way of allowing anyone who wants to either decrypt and/or authenticate the email as having come from me, I need some way of allowing them to have access to my public key. Well, we could use DNS to distribute public keys, but can't until it's absolutely securable, until we absolutely know that this is my public key, and that that cannot be spoofed. Otherwise the damage would be too great.

So we have a problem, though, in deploying DNSSEC. Basically the technology has largely been resolved. We have a system now which looks like it'll scale. The problem is that, in the scenario I just painted, we are needing parents to sign the children's hash, that is, the children produce a signature of their current domain and get the parent to sign it. Well, that means that the parent is using its private key in order to sign data, and that the parent's public key is made available. Well, that's the way public key infrastructure technology works. The problem is the politics of this because, in order for GRC to be authenticated, there has to be a chain of trust all the way up to a trusted root, what's called a "trust anchor," some ultimate authority from which all authority emanates, because that root, that is, literally the root servers would have to sign all of the top-level domain, all of the .com and .net and .org and .mil and .gov, and the other countries' top-level domains, like .it and .se and .ru and, you know, and all of the country code domains similarly would have to be signed.

So the question is, who gets that? What authority runs that top-level root server key? And unfortunately that hasn't been an issue until now. But the U.S.'s Defense Department and Department of Homeland Security have said they want to be in charge of these keys. And unfortunately the...

Leo: I don't think Russia will like that.

Steve: There's a problem, yes, there's a problem with the rest of the world swallowing the U.S.'s unilateral management and ownership of the keys.

Leo: This has been going on in other areas, as well, with ICANN, for instance, for a while. I can remember last year there was a move in the U.N. to take responsibility for the domains out of the hands of the - essentially out of the hands of the U.S. Commerce Department. And now it's even worse if they want DHS to do it. But there's some resentment among countries like Afghanistan, they're saying why should the U.S. be responsible for this? Of course we're responsible because we

invented it. But now it's a global system.

Steve: Right. And so, well, exactly, so there's a bit of a chicken-and-egg issue here. It's like, well, until it became really valuable, no one really cared.

Leo: Right. Yeah, we did it. Now you want it from us; right.

Steve: Yeah. To give you some sense for the complexity at, I mean, the way things are even now, that is, how the whole root zone system operates, I can read a short paragraph that talks about the issue of securing the root. And it says, "To clearly understand how this proposal will impact modifying and publishing the so-called 'root zone file'" - so there's a single file, ultimately a single file called the "root zone file." And that contains the addresses of all the top-level domain name servers, that is, all the .com servers, all the .org servers, all of the .edu servers, .gov servers, all of those, and all of the other country code servers are in this top-level zone file. And understand that, okay, remember that what this is, is this is the IP addresses of the name servers of all of those top-level domains. And that's in a single file called the "root zone file," or RZF.

So continuing, it says, "So to clearly understand how this proposal will impact modifying and publishing the root zone file, it is best to briefly outline the current process and actors involved. Currently, change requests from registries are sent to ICANN, specifically to IANA, for processing. Once it is determined that the changes meet IANA's narrow technical requirements, and they are approved by the ICANN board, the request is forwarded to the U.S. Department of Commerce for review and approval. If the Commerce Department approves, the root zone maintainer, the RZM, currently VeriSign, edits and generates the revised RZF, the root zone file. The root zone file is then loaded by the root zone distributor, the RZD, also VeriSign at this time, to the distribution master name server. Once there, it can be retrieved by the other root server operators located around the world."

Leo: Sounds like something out of "Tron."

Steve: It's, well, okay. This is the simple way that it is now. And so the problem is that implementing DNSSEC, DNS security, requires modifications to everything. I haven't even talked about the details of the technology of this. But there are real problems. For example, I've sort of talked glibly about digital signatures and keys. There are key-signing keys and zone-signing keys, and another record called a "delegation signing record." I mean, it gets very complicated. Also, unfortunately, it gets bloated. For example, in an example I saw, a normal, traditional DNS record was 75 lines long. And that's a relatively - that's a reasonable zone file, 75 lines long and about 2.3K in size. The same zone implemented with DNSSEC was expanded from 7[5] lines to 665 lines, and from 2.3K to 27K.

Leo: But that's still small. I mean, we...

Steve: Ah, except, Leo, that it no longer fits in a UDP packet.

Leo: Oh, that's not good.

Steve: And that's the problem is the original DNS spec specifies that UDP DNS messages can be up to 512 bytes of payload, which ends up being 576 bytes for the total packet. So 512 bytes. Now, 512 bytes doesn't sound like a lot. But it turns out because of the way DNS messages are compressed - there's a compression technology which is pretty clever that just uses simple pointers to end up compressing a very complex set of queries and replies into a couple hundred bytes. So it ends up being that UDP is entirely practical, single packets, for shooting this stuff back and forth. Well, suddenly, when you add DNSSEC, that is no longer the case.

Leo: But what's wrong with sending multiple packets? Most of our data is multiple packets. Does that screw something else up?

Steve: Well, okay. There's no provision, first of all, for multiple UDP packets. Remember that it takes TCP to create the notion of a byte flow which can then flow across multiple packets. So DNSSEC requires other enhancements to DNS which are identified and have been put in place, and that is extensions that allow DNS to use UDP over much larger packets. So that's been done. The problem is you'll remember that, when we talked about this before, the way the existing DNS system is functioning, it is more than half saturated, meaning that all the servers that are currently in use, they have less than double capacity available. And DNSSEC hugely is more than...

Leo: Well, it's a hundred times bigger.

Steve: It's, yes, hugely more than what we have now. So this, I mean, it really does represent a massive burden to the infrastructure.

Leo: Does this mean it's not going to happen?

Steve: I mean, it's been trying to get off the ground now for years. And it's having, I mean, this gives you a sense, as I said, I hadn't even talked about the technical side of this, which is really a problem. I mean, this really represents problems. Also we've got suddenly public key operations going on which we know are expensive. All of this signing and verifying, these are public key transactions which are computationally burdensome. One of the problems with DNSSEC is that now it becomes much more feasible to launch a Denial of Service attack against DNSSEC DNS name servers. It used to be that you would flood them with data. Now you can lead them on a merry chase of trying to resolve DNS records, and they'll collapse because of the computational burden of all the public key work that they have to do.

Leo: So you've got the geopolitical issue, you've got the technical issue on the one side pushing against it.

Steve: And the performance issue and the bandwidth issue.

Leo: Performance and bandwidth pushing against it. On the other hand, you've got the whole world saying, but we need a secure DNS system.

Steve: Yes, well, exactly. We need - imagine having a truly robust, secure, hierarchical database so that you could distribute, not just IP addresses, but as I was saying before, even securely distribute public key certificates. Suddenly then, like who knows how long they would last, but I could have a public certificate that anyone, through a simple naming mechanism - and DNS names are a naming mechanism. Through a naming mechanism it would be possible for anyone to acquire anyone else's public key and then be able to use that for any number of purposes. It would be really tremendous.

Leo: So what do we do? Lobby our members of Congress? I mean, what can you do? Who's in charge here?

Steve: They recognize that there is a chicken-and-egg problem, that there would be a lot of demand for this if it existed. But there isn't demand for it now because it doesn't exist. There are...

Leo: Who implements this? Does my ISP implement this? Do the backbone servers implement it? Does the government? Who would be the decision-maker?

Steve: Well, and that's the other problem is that, in order for this to function, every name server, every server in the chain of command, has to have an - there has to be an unbroken, essentially unbroken zone signing from the zone you want to verify all the way up to the root.

Leo: And you know how well it worked when we moved to IPv6 and everybody just jumped on that bandwagon.

Steve: Exactly.

Leo: Clearly this isn't going to happen, not for a while.

Steve: Exactly. And, now, there is sort of a funky alternative approach which - and the acronym is escaping me right now. We're in acronym soup today.

Leo: I think we talked - did we talk about it when we talked about this?

Steve: No, it's DLV. Oh, it's a lookaside approach. The "L" stands for "Lookaside." The idea being that ideally you'd like to have a single coherent hierarchy where at every level in the hierarchy you have keys, and you're able to verify - by simply knowing the public key of the root, you can then verify the entire hierarchy yourself, just like by knowing the public key of a certificate authority you can then verify the entire chain of authority all

the way down. However, it's recognized that this effort is stumbling because of the issue of the requirement to have one single central authority, and no one can agree geopolitically who that authority is. So there is a counterproposal that allows the so-called "lookaside" technology, where somebody like the ISC, they would maintain their own servers. And so GRC would develop its own public key and private key pair, and we would provide, securely provide the information to ISC, that is, our public key to ISC. And then GRC would use DNS naming. For example, it would be GRC.dlv.isc.com. And so any domain could be subsidiary to ISC - I'm sorry, to ISC.org, could be subsidiary to dlv.isc.org. And that would allow someone to, first of all, then you would only need ISC's public key in order to verify their signature of GRC's public key in order to get GRC's public key to verify GRC. So basically it's a mess.

Leo: It's a lookaside.

Steve: It's a lookaside, or lookaskance. It's a non-hierarchical alternative to the hierarchical approach. Not as good; but, I mean, still gives you provability.

Leo: Would it be more robust? Because the problem with the hierarchical approach, as we've talked about before, you knock out the top, everything collapses.

Steve: Well, the benefit of the hierarchical approach is that it scales.

Leo: It's efficient, yeah.

Steve: Yeah, the problem here is that ISC's servers would come under tremendous burden because they would have to field a query from everyone who wanted to verify any domain...

Leo: Oh, so it's even more vulnerable because, again, you have that choke point.

Steve: Yes. Yeah. Well, it would be bandwidth vulnerable. They would be signing, for example in the GRC example, they would sign our public key once and then maintain that as a static record. So anyone would have to have their public key, and then anyone wanting to verify that would have to be doing a public key operation. But that would still be the case even in the hierarchical arrangement.

Leo: You wouldn't need - would you need fewer check-ins with this system? I mean, does the key stay good? Or do you still have the same time-to-live issues where you have to update it regularly?

Steve: Ah, as a matter of fact we didn't talk about that. But you have - you still have TTLs. But then you also have relatively short lifetimes on the keys. There are two classes of keys. I mentioned key-signing keys and zone-signing keys. It turns out that some of them, for security reasons, have to have lifetimes as short as 30 days. So then you've got to be rolling keys over where you have replacement keys that overlap the valid times

of the keys they're replacing. I haven't gone into the detailed technology of this because, I mean, it's really, I mean, even what I've described is rather mind-numbing. But the technology of this is even more so. But they've worked out the details, and it exists, and it's clearly and cleanly described in RFCs.

Leo: Is there a likelihood that either of these systems will be implemented in the near future? I think we're just going to stumble along with what we've got.

Steve: There is one alternative approach that has been proposed.

Leo: Oh, boy.

Steve: Yes. Which is...

Leo: Another one.

Steve: Which, well, it involves not relying on a sole signer of the root. Because remember, and we've seen this before, it's possible to have multiple signers of a single object. And then you're not - then you're able to decide...

Leo: Ah, I like that.

Steve: ...whom you want to trust.

Leo: Right.

Steve: Okay. You like it.

Leo: That's kind of like PGP.

Steve: Exactly. So here it says - again I'm reading from this paragraph because otherwise there's no way to explain it.

Leo: Well, this could solve a lot of things because you could have - Russia could sign as well as the U.S.

Steve: Yes.

Leo: You don't have this single point of failure.

Steve: Yes. "Instead of a single RKO" - that was the root key operator - "we propose that multiple independent non-governmental RKOs be responsible for generating KSKs" - which are the key signing keys - "and ZSKs" - which are the zone signing keys that I've just talked about. Just believe, you know, just for now just take it as a fact that they exist.

Leo: Yes.

Steve: "...and transmitting the public portions of these keys to the root zone maintainer for construction of something new, a root key set. The RKO, that is, the root key operators, would be also responsible for distribution of the public portion of their key-signing key, which is the trust anchor globally. So once constructed, the root key set would be distributed to the RKOs, the root key operators, for signature over all the key sets. The signed root key set is then returned to the RZM, which is the root zone maintainer, for inclusion in the RZF, the root zone file. Each root key operator will then sign a copy of the root zone file using the private portion of their respective zone signing key and transmit it back to the root zone maintainer, who will merge the files."

Leo: Certainly.

Steve: "All of the exchange" - you knew that was coming.

Leo: Of course.

Steve: "All of the exchange of data between root key owners," I'm sorry, "root key operators and the root zone maintainer would occur on secure, out-of-bound channels." Anyway, the upshot of this is that we would end up with a master root zone file that had been essentially signed by as many participants, nongovernmental participants, as wanted.

Leo: Yeah.

Steve: And then you could use any of those participants' public keys to verify the integrity of the globally master root zone file.

Leo: Even the Hong Kong Post Office, if you will.

Steve: And then no longer be forced to trust, for example, just in any single entity.

Leo: I like this. This has worked very well for PGP. When you create a PGP key, in fact, people even have key-signing parties. You ask people who know that this is you to sign the key and to add trust to that key. It's a trust model. And I think it works very well. And of course you decide as the user - now, end users aren't going to know who to trust. End users are going to make that decision, Microsoft's going to make that decision, or Safari, or Firefox. But they would probably have, in the same way they do now, right, a set of trusted key signers. Right?

Steve: Yes.

Leo: I think that that seems like a good system. That seems like it's implementable. It distributes the burden. It's global, not U.S.-centric. Is this kind of a favored system? It seems like one that could actually work.

Steve: Yes. And that's the direction that we're heading in because, I mean, for now, for a couple years, the problem of who's going to be the sole arbiter of the root has just been a nonstarter. I mean, it's just...

Leo: It can't be the U.S. It just can't.

Steve: Yeah. And we're not wanting it to be anybody else.

Leo: Right.

Steve: So that kind of multiple overlapping signatures is probably the way it's going to evolve. Right now there are some countries that have their own top-level domains signed and are using DNSSEC. The problem is that it's not global, that is, in order for me to trust, or any name server to trust any of their children domains, for example, underneath I think Sweden is .se, and I think Sweden does have DNSSEC running at their top level. So for any of the domains under - for an external name server, random company, random ISP in the U.S. to trust any of the domains underneath .se...

Leo: They'd check with the domain server, the key server at domain .se, the Swedish...

Steve: Exactly. They would need to have .se's public key, that is, they would have to know it. And they have to securably know it. That is, spoofing that becomes a big problem. That's what you have to absolutely prevent. So you need a secure way to get the public key of the .se server. And the problem is, and any other servers that are not the root. So it makes much more sense if everything descends from the root. Then all you need is one public key that could be globally known, and some mechanism for updating that periodically because you don't want to have to have it last forever.

Leo: Okay, good.

Steve: So those are ultimately...

Leo: Sounds like we've got something that could actually be workable for all parties.

Steve: It could be workable. Don't forget, though, we've got computational problems. We've got DNS server deployment problems. We've got bandwidth problems. I mean, there's...

Leo: So this would sit on top of a DNSSEC solution that would have to resolve the UDP/TCP issue, that would have to resolve the bandwidth issues, things like that.

Steve: Bandwidth and computational burden of essentially moving - and for me it's phenomenal to look at how simple the system we have now is. And what you have to do...

Leo: It's amazing that it's worked for so long.

Steve: Yeah, I mean, it does.

Leo: You and I have talked about this vulnerability, we don't talk real loud about it, but the fact that these 13 root servers, if they could be DDoS'd for a day, the whole 'Net would go down.

Steve: And they've been DDoS'd for short periods of time, and it's a problem. But that has been - they are actually not physically 13. There are 13 IPs, but they've got much more hardware behind them. And that infrastructure has been hugely strengthened over the years so that it's much more difficult to actually mess with them now. But it's a perfect point is that, for example, I'm running my own independent resolver here. So I'm not depending upon any ISP's resolver. And my local DNS resolver knows those 13 IPs. Those are the 13 root server IPs. And every so often, not often, it checks in with them to update the record of the top-level domains in order to get that from them. And so the same sort of thing has to happen in DNS.

And as I was saying before, it's remarkable to me how bad it gets when you really want to apply security to this system. I mean, it's so simple, relatively, to have DNS work the way it does in this sort of "trust everyone" model and give me an IP for a domain. When you really, really want it to be cryptographically secure, it's incredibly burdensome for that to be added to the system. But, boy, with the chicken and egg, you know, once we do, we get something really valuable. We get a phenomenal ability to have a globally distributed, secure database. And ultimately I think it's clear that's where the Internet's going to have to be.

Leo: We've got to do that. I mean, we can't not do this. Well, this has been a very interesting conversation. Of course a difficult one to follow. But, as usual, you do a great job explaining it. And I'm watching the people in the chatroom going, what's he say? What's going on? What's it all mean? But we figured it out, I think. Hey, Steve, thank you so much for making that clear. And I understand, you know, somebody in the chatroom did mention that the U.S. government has said it's going to deploy DNSSEC, I mean, that that is - that the White House says we're going to do it.

Steve: Well, .gov and .mil, I know .gov, I think .mil, those two top-level domains will. But again...

Leo: They can do that unilaterally because those are both U.S. domains.

Steve: Yes, but not the root. And so there again, anyone who was - anyone within that system would have their servers configured with the public key of the .gov domain and the .mil domain and would then be able to authenticate any records coming from within that structure, but not outside that structure.

Leo: Thank you, Steve Gibson. You're the man. If I ever need anything like that explained, I'm going to you.

Steve: Or just tune in next week for...

Leo: Another thrilling edition. Next week we'll answer your questions. And again, let's underscore this, GRC.com slash...

Steve: Feedback.

Leo: Feedback. GRC.com/feedback. Hey, while you're at GRC check out of course SpinRite, the best, the one, the only - I have now a policy that every hard drive that gets deployed at TWiT gets SpinRited before it gets deployed. And that just gives me a nice feeling to know that we're going to map out bad sectors. We're going to have much more reliable drives. And since we're recording all the video now on a new drive every week, I think it's fairly important to do that. GRC.com. Get your copy of SpinRite, the world's best disk maintenance and recovery utility. Also while you're there, lots of free stuff. ShieldsUP!, lots of free programs like Wizmo. And let's not forget 16KB versions of this very show, show notes as well so you can follow along, and transcripts, too, thanks to Elaine. It's all there. GRC.com. Steve Gibson, thanks for joining us.

Steve: Talk to you next week, Leo.