SECURITY NOW!

Transcript of Episode #276

Testing DNS Spoofability

Description: After catching up with the week's security updates and news, Steve and Leo revisit the continuing concern over DNS Spoofing by examining the technology behind Steve's quite comprehensive, free, online DNS Spoofability Testing system at GRC.com.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-276.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-276-lq.mp3

Leo Laporte: This is Security Now! with Steve Gibson, Episode 276, recorded November 24, 2010: DNS Spoofability Test.

It's time for Security Now!, the show that covers your privacy, your security, everything you need to know to keep yourself safe online. And this is the guy who does it, the guy to my left, Steve Gibson, the man behind GRC.com, SpinRite software, and a whole bunch of really useful utilities, including that DNS Benchmark that we talked about a couple of weeks ago. Hey, Steve. Good to see you again.

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

Leo: Happy Turkey Week.

Steve: Yes.

Leo: What are you doing tomorrow?

Steve: It's T-Day. I have a few bachelor friends who are, actually, they're cooks, and I'm not. And I'm not even going to supervise. I'm just going to stay out of the way.

Leo: Just show up, that's the best.

Steve: And none of us are going to deal with our families tomorrow. We're just going to get together and make a nice meal.

Leo: That's wonderful.

Steve: So it'll be very nice, and we'll deal with families on Christmas.

Leo: I'm leaving as soon as the shows are done today. We're going to drive down to Santa Cruz, where my father-in-law lives, and we're going to stay in a nice little hotel down there, and we're taking them to the hotel's restaurant for Thanksgiving. So that'll be fun.

Steve: Simple.

Leo: Simple.

Steve: Perfect.

Leo: No dishes to clean. The only negative, no turkey or stuffing leftover.

Steve: That's true.

Leo: That's my favorite part. I love those turkey sandwiches.

Steve: There will be sandwiches for us. We have a 14.28-pound bird.

Leo: Wow. So, now, what is our topic of the day today? I think we have something kind of interesting to talk about.

Steve: Oh, I think everyone's going to find it interesting. We talked two years ago, actually about two and a half years ago, about the problem that Dan Kaminsky brought up when he realized that a huge percentage of the Internet's DNS services, the domain name servers, were much more vulnerable to spoofability than people recognized. Around, I guess it was that summer, while this was being unveiled, around the Black Hat conference, I realized that, in a vein very much like ShieldsUP!, which has been around of course forever...

Leo: And that tests your router for holes.

Steve: Well, yeah. Actually it tests your connection. So it's GRC's server. It's a free service that of course GRC.com makes available where you inherently need something on the outside to be probing back towards you. So I realized many years ago that it was a service that a website or a web server could offer. Similarly, I realized that it would be possible to create a service that tested the spoofability in exactly the way that Dan Kaminsky understood it, and the way the industry came to realize, oh, my goodness, 25 percent of the DNS servers on the 'Net are currently very spoofable. I realized that, similarly, a service could be created to check the DNS servers that people were currently using.

So in short, you could go to GRC.com, click a button, and my servers would check your DNS servers to see how spoofable they were and then produce a report on the fly, much as ShieldsUP! does, which provides you a complete diagnostic analysis of the way your DNS servers are behaving relative to spoofability. So I set about creating such a facility back then. And I've had various things that came up in the meantime, the Benchmark, and of course I got all entranced with PDP-8 technology, remember the old computers, and built...

Leo: They're right there behind you, over your shoulder, yeah.

Steve: Flashing behind me and so forth. And then of course the Benchmark that we talked about two weeks ago fit into this because it was about DNS. And so essentially this is the other side of that. The Benchmark is something you run on your computer to help you choose the right DNS servers. But after having done so, you want to make sure that the result of that choice is not spoofable. So then you use GRC's free DNS spoofability testing system to analyze the queries that those DNS servers you've chosen are emitting out on the Internet. So today we're going to talk about how it does that.

Leo: Sounds great. Sounds very cool. I'm kind of excited. All right, Steve, I'm ready to start. We will start first with, what, security updates, yes?

Steve: Yeah, we've got some updates and sort of some news, or some update news. It's been a very quiet week. The only thing that happened was that Apple did update Safari. When I turned my Mac on this morning it said, oh, we've got a new Safari for you. I think it was 37MB. So we're now at v5.0.3 for both the Windows and the Mac platforms. They fixed a bunch of vulnerabilities. I saw the number 27 somewhere. So they cleaned things up. There were some things that were fixed also in Chrome's browser, I guess due to the WebKit commonality, that is, that they both use some of the same core libraries from WebKit. I had read that four of the 27 flaws had been fixed in Chrome already. So Apple was catching up and fixed 27 things. So Safari is updated.

Also, just yesterday, the EFF introduced a new updated version, still at beta, I think it's 0.9.0 is their version, of their HTTPS Everywhere add-on for Firefox. That's something which it's a little tricky because it involves some customization per site. As we've talked about before, it's not possible to simply force every website to use SSL connections, that is, HTTPS connections. However, many sites which don't explicitly do so, can do so.

So what the HTTPS Everywhere plug-in for Firefox does, and this was specifically updated to deal with Firesheep, which we've talked about, which is of course the non-HTTPS sniffing, session-hijacking add-on which has now been around for a month and a half or so, which continues to be downloaded, and people are getting up to no good with it. It was created, as we know, to highlight the vulnerability of people using insecure websites which maintain their logon state in an insecure fashion. So HTTPS Everywhere - which is a free download from the EFF, you can just put in "HTTPS Everywhere," and Google will take you right to it - now has been enhanced to support Google Search, Wikipedia, Twitter, Facebook, bit.ly, GMX, Wordpress.com blogs, The New York Times, The Washington Post, PayPal, the EFF's own site, TOR, Ixquick, and many others.

Leo: That's most of the stuff Firesheep does; right?

Steve: Yes. And so they literally, they went after those things that Firesheep was doing. And they specifically worked on those sites which were vulnerable. And so the idea is that you just can't force, you can't just stick an "S" on the end of all the HTTP's on a given site. But individual sites can be customized with some scripting which the HTTPS Everywhere comes with and which individuals are also able to customize, in order to sort of understand the URLs, maybe look at patterns which can be moved over to SSL in order to increase the security. So it is something that I recommend without hesitation to any user of Firefox. Add this, and it'll just add security where possible, where it's known to be possible, on all of the sites that it supports. And that's a growing list. So this is something that is very nice.

Leo: Yay. Thank you, EFF.

Steve: Yes. Adobe just took their PDF Reader to version X.

Leo: Because they hit DEFCON 3.

Steve: Burying it deeply would be nice. So they had been at v9. We've talked about X several times, saying that it was coming, and that we expected it sometime before the end of November. So that did happen. What's significant about this is that they've added sandboxing. So Adobe Reader now has sandboxing. It won't solve all the problems. We will continue to have problems. But it does raise the bar. And it will probably mean that, in the future, the rate of problems hopefully is lower. Although I got a kick out of you, prior to us starting to record, Leo, one of the problems that you had run into on the system was that you had found that something had installed McAfee's Secure Scan.

Leo: Oh, yeah, I was really mad about that. I think it was Firefox, wasn't it?

Steve: It may have been.

Leo: Because they give you little stuff, you know. And if you're not careful during the install and don't uncheck the boxes...

Steve: Yup. And when I went to Adobe's site, which is just get.adobe.com/reader, again, to move up to Adobe Reader X, which they use a Roman numeral "X," it's get.adobe.com/reader. In the upper right-hand corner, checked by default, is a sneaky little checkbox for "McAfee Security Scan Plus"...

Leo: That's probably what it was. It was Adobe.

Steve: ...to be included in the download. And I also noted that this thing includes Adobe AIR.

Leo: Well, I'm annoyed because I don't know who put - I didn't want that on my system. Not McAfee. I'm not worried about McAfee. Get Adobe off of my system. We use the other one, that I like much, much better anyway, Foxit.

Steve: Yes, Foxit. I was going to say that, with this getting just bigger and bigger, and now with them dragging Adobe AIR along, which many users will not need, and they're just doing it because they want to establish that platform on more people's machines, and that will of course induce developers to do AIR-based things. It's really long past time to look at a different reader, I think.

Leo: Foxit's quite good, I think.

Steve: The PDF format is now universal enough that it's not as if you're going to have big compatibility problems if you don't use the original solution. And the other readers are just not going to be as large a target. We don't know that they have fewer security problems. We do know that fewer are being reported, and fewer are being exploited, because Adobe has the big bulls-eye painted on it. So it really looks to me like it makes sense to use something else.

In the news, Washington Post covered a story that began with - this was on Stuxnet. "A malicious computer attack that appears to target Iran's nuclear plants can be modified to wreak havoc on industrial control systems around the world, and represents the most dire cyberthreat known to industry, government officials and experts said Wednesday." So this was some testimony in front of Congress about Stuxnet and what it means to everybody else. Of course we now know, as we talked about last week, that it really does appear that Stuxnet was designed deliberately to mess up the centrifuges being used to enrich uranium for Iran's - hopefully only their commercial nuclear reactor projects and not for bomb-making, not for taking it to a weaponizing level.

Leo: Maybe we could do this to North Korea.

Steve: Yeah. And so what the testimony basically did was serve to pretty much horrify members of Congress about the concept that Stuxnet could now be and would be reverse-engineered. Whereas it had been targeted specifically to Iran's process control systems for nuclear enrichment, nothing prevented it from being retargeted at other segments of the process control industry - power stations, nuclear reactors, all kinds of things that are using Siemens-based control systems. I mean, basically the template is all there now. And reverse-engineering is possible, and it is entirely foreseeable that many of the tricks that Stuxnet pioneered, I mean, now that it's spread, and now that everybody who wants a copy has a copy, it's certainly possible to retarget it.

So it's looking like this may result in some legislation from the government beginning to enforce a wall, essentially, I would use the term "firewall," the idea being the separation of offline process control things from sort of online, civilian use, Internet-connected stuff. What everyone is recognizing is that it's the sort of lazy, oh well, it's like the computers in the ER, in the operating room theater in hospitals being on the Internet just because it's convenient. It's like, oh, well, they get their updates and so forth. Well, we've seen what happens when operating theater, hospital-based computers get infected with malware. It can literally be life-threatening.

Well, this is the same sort of problem where Stuxnet is crossing from the Internet and machines that are on the Internet into process control systems. The only way that's possible is if there's connectivity. So it's looking like the government is beginning to say, hey, we need to do something about these critical infrastructure systems which are on the 'Net. They really need to be taken off the 'Net. Yes, it's not going to be as convenient for people not to have all of the PCs that are running Windows hooked to the Internet. But if they're in charge of running nuclear reactors, we really don't want the rods pulled.

Leo: Please.

Steve: Please.

Leo: Please.

Steve: And also in the news, I thought it was interesting, and a number of people wrote in. There was some news about China having last April briefly rerouted 15 percent of the Internet's traffic through their country.

Leo: That's weird.

Steve: Well, we'll be covering this in detail, that is, the technology for this, when we get into our forthcoming, and still forthcoming, basics of how the Internet works series, where we're going to start right back at first principles, bits and bytes, and work up to the whole Internet and how it functions. It actually wasn't 15 percent of the Internet's traffic. It was 15 percent of the Internet's routing network prefixes.

Essentially, we know generically, sort of in an overview, that the Internet is made up of routers scattered far and wide that are interconnected to each other; and that packets put onto the Internet are then bounced from one router to another, each bounce taking them closer to their destination. Well, the way that's done is that the routers contain multiple interfaces, each interface connecting it to one other router. They have multiple interfaces, so each router is typically connected to many other routers. And routing tables in the router, when a packet comes in, the destination IP address is examined. And it's examined against this routing table which tells the router which is the best direction to send that packet towards its destination.

So there's a protocol called BGP, Border Gateway Protocol. It's the protocol which routers use to communicate among themselves who has the best routing direction for given packets. Well, there's been some problems in the past. BGP uses the TCP protocol. And some of the early routers had easy-to-guess sequence numbers in their TCP communications, which allowed TCP connections to be hijacked and spoofed, and allowed bad guys to poison the routing tables of routers, causing traffic to get misrouted.

Well, it also is the case that mistakes can be made. And it's most likely that someone in China made a mistake and essentially published, because that's the term that's used, published the incorrect information that particular routers in China were the best place to send packets, like for U.S. .gov domains. And it turns out that many very sensitive domains were, for the period of about 15 minutes, were routed through China. And so what happened was this information got published. And as happens with routers, they propagate and update each other.

So this misinformation propagated through the routing infrastructure of the Internet, causing all the routers everywhere to say, looking at their packets, just, like, "Oh, apparently China is where we want to send this." And they did. When China routers got it, they thought, I mean, they saw the packet coming in and thought, wait a minute, we don't know what to do with this. This needs to go over there. So they dutifully bounced it back out to wherever the packet was bound, many of them bound for sensitive domains in the U.S. And an analysis of this really makes it look like it was a mistake more than it was deliberate.

But it certainly is an interesting warning to people that what can be done by mistake can also be done on purpose. And this is yet another reason why encryption is a good thing to have on all your communications, because, literally, unencrypted traffic was all sent to China for a while before it got sent back where it should have gone. Which raises the hairs on our Homeland Security folks, who watch these things.

In other news, our friend Charlie Savage at The New York Times, who reported on the FBI's increasing interest in wiretapping - which raised the hair on the back of my neck because of my interest in doing a highly secure, encrypted VPN product that we've talked about, CryptoLink is what it will be called. He updated with an interesting story recently. And I'm just going to quickly read this rather than trying to paraphrase it. He wrote that:

"Robert S. Mueller III, the director of the Federal Bureau of Investigation, traveled to Silicon Valley on Tuesday to meet with top executives of several technology firms about a proposal to make it easier to wiretap Internet users. Mr. Mueller and the FBI's general counsel, Valerie Caproni, were scheduled to meet with senior managers of several major companies, including Google and Facebook, according to several people familiar with the discussions. How Mr. Mueller's proposal was received was not clear.

"'I can confirm that FBI Director Robert Mueller is visiting Facebook during his trip to Silicon Valley,' said Andrew Noyes, Facebook's public policy manager. Michael Kortan, an FBI spokesman, acknowledged the meetings but did not elaborate. Mr. Mueller wants to expand a 1994 law, the Communications Assistance for Law Enforcement Act (CALEA), to impose regulations on Internet companies. The law requires phone and broadband network access providers like Verizon and Comcast to make sure they can immediately comply when presented with a court wiretapping order.

"Law enforcement officials want the 1994 law to also cover Internet companies because people increasingly communicate online. An interagency task force of Obama administration officials is trying to develop legislation for the plan and submit it to Congress early next year." That would be early in 2011. "The Commerce Department and State Department have questioned whether it would inhibit innovation, as well as whether" - in my case it would.

Leo: Yes. Yes, the answer is.

Steve: "...[A]s well as whether repressive regimes might harness the same wiretapping capabilities to identify political dissidents, according to officials familiar with the discussions."

Leo: Some might say it turns our own regime into a repressive regime.

Steve: Uh-huh.

Leo: But that would be a cynical point of view.

Steve: "Under the proposal, firms would have to design systems to intercept and unscramble encrypted messages. Services based overseas would have to route communications through a server on United States soil where they could be wiretapped. A Google official declined to comment. Mr. Noyes said it would be premature for Facebook to take a position." So as our listeners know, I will be watching this and reporting on this as we move forward.

Leo: You know, Dvorak has always said that Facebook is a CIA front. And it's a very flimsy thing. But the CIA is a big investor in Facebook. Did you know the CIA has a venture fund?

Steve: Okay, that's annoying.

Leo: Yes [laughing]. The CIA has a venture fund and is in fact a Facebook investor. Now, I mean, if you really want to be a conspiracy theorist, what better way to surveil people than Facebook? Because they voluntarily give you everything.

Steve: In fact, I was listening to your live feed while things were getting set up for us recording this show. And one of the news blurbs that Tom brought up, Tom and Becky mentioned, indicated that insurance companies, life insurance companies, were now going to be increasingly using Facebook to check on publicly available information which people had posted about themselves in order to get information about them which would then be used in some fashion to further profile them for life insurance purposes and health insurance purposes.

Leo: See, that's really who's most interested, I would guess.

Steve: It all gets kind of creepy.

Leo: I don't know if it's a conspiracy theory or if it's real. But it's just something to be aware of.

Steve: Yeah. So we know that 64-bit versions of Windows have been designed from scratch to be dramatically more secure against malware of all kinds, and specifically against rootkits. We've talked about how Microsoft has this dilemma, that it is difficult for them to immediately secure 32-bit versions of Windows because they can't impose the same level of security without breaking things. 64-bit Windows thus, until now, has been virtually immune to really nasty attacks like rootkits. Well, the Alureon rootkit, which has until now only been found on 32-bit Windows, has now successfully compromised 64-bit Windows. That was the rootkit which we did discuss earlier this year, I think it was, when a version of Microsoft's updated software caused crashes on 32-bit Windows, just regular Windows Update, which ran, and suddenly people couldn't boot their Windows anymore.

Well, that was caused by the fact that their machines had already been infected with a 32-bit version of this Alureon rootkit which had infected their boot sector and was assuming hard-coded entry points into Windows kernel. The update patch from Microsoft changed those hard-coded entry points, causing Windows to no longer boot because the rootkit was crashing Windows; whereas before that it had been carefully designed not to do so. So 64-bit Windows has specific, rigorous, driver-signing protection and kernel PatchGuard protection, all designed to make it bulletproof. The problem is, if something is running before Windows, and if it's cleverly designed, there's nothing Windows can do, literally nothing Windows can do to protect itself.

And that is exactly what has happened here. The security people who have really looked at this understand this problem. That's why the Trusted Platform Module (TPM) is called the Trusted Platform Module. The idea behind TPM - which is still not fully implemented. We have the hooks, we have the hardware, we have the technology. But it hasn't yet happened. The idea behind that is that from the first moment power is turned on, there is a verifiable protected environment that, step by step, cannot be compromised. It's because we don't yet have that, that it is still possible for the boot sector of the hard drive to represent a point of compromise.

And so it is this altered boot sector which gets control as the system boots before Windows does. The BIOS turns over control to the boot sector, which runs some code that is able to find the rest of the rootkit, and on the fly it patches the kernel protection to disable it, allowing an unsigned device driver to get into Windows, which then flips a switch, which is a development time switch used to bypass driver signing in 64-bit Windows, and then your 64-bit, much-more-hostile-to-malware Windows is rooted. And it's happening today. So just FYI [laughing]. The good news is that...

Leo: Sorry, I was just running in the other room for a moment. That's very scary. But didn't Microsoft say that the 64-bit version of Windows was, like, preliminary to a fully secure version, but just kind of a warning shot across the bow of companies? I mean, this is something they'll fix, presumably.

Steve: They know about it. Their various security tools are aware of it. But essentially what it means is that this notion that 64-bit Windows is impervious, I mean, it is as impervious as they were able to make it. The point was...

Leo: Without breaking everything.

Steve: No, no, no. That was the 32-bit problem. They were able to start fresh with 64-bit Windows. And they did say that PatchGuard is here, drivers must be signed, address space layout randomization is here, I mean, basically they were able, because they were starting late in the game on 64 bits, they were able to do everything they possibly could. And so while doing everything they possibly could, they've been rooted. And so that's significant.

Leo: I do remember, though, there were workarounds for PatchGuard even when they first announced it.

Steve: Okay. So here's the problem. This is software. There will always be workarounds. It's spy vs. spy. You've got other software fighting against Windows. We're never going to have perfect security on an open platform where users can install software which they download from anywhere on the Internet. We're just never going to have it.

Leo: That's nice.

Steve: Also in the news, Google quickly fixed a little glitch in their Google Apps scripting API. A 24-year-old Armenian who was using the handle "Vahe G," created a demo on Google's own Blogspot blogging platform. And actually it probably had to be on Blogspot because visitors who went to his blog page who were sort of simultaneously logged onto Google accounts, so like if you were using a Gmail account, or you were a Gmail user, and you had, like, checked the "yes, keep me logged on" so that you could go back and still be logged on, his little demo used a mistake in Google's scripting API to obtain the email addresses of anyone who went to his blog page, despite the fact that they hadn't given it to him, and sent email to them from Google with fully valid Google headers.

So this quickly came to Google's attention. They removed the page, and then they fixed the little flaw that he had found. But it generated a lot of news in the security community because it was like, whoa, whoops, wait a minute. My Gmail address has been leaking. I mean, essentially, he could have been harvesting them and using them for whatever purpose he wanted. So that was fixed.

And lastly, our old friend Phorm, which we gave a whole podcast to a couple years ago, this was the very nasty technology which, I think it was BT, British Telecom in the U.K. was testing without its customers' knowledge. It used equipment installed at BT to, in real-time, intercept their customers' communications and install cookies, Phorm cookies, on every single domain in their browser that they went to in order to tag them and track them and profile them deeply and pervasively. This is the so-called Deep Packet Inspection. And we, hopefully, had thought we'd seen the last of this. The bad news is it's trying to make a comeback. The Wall Street Journal in their ongoing series, "What They Know," on Internet spying, had a story titled "Shunned Profiling Technology on the Verge of Comeback." And now Phorm...

Leo: Like a zombie.

Steve: I know.

Leo: The undead.

Steve: It will not die. Now they're saying that they're going to opt in, rather than opt out, because you could use an opt-out cookie, except that doesn't work because no one does that, no one knows about it. And apparently now you'll be able to opt-in, and users - get this, Leo - can pay $10 a month not to be profiled and "Phormed."

Leo: Well, that's not opt-in, that's opt-out.

Steve: Yeah.

Leo: I don't understand. If you have to pay not to be profiled...

Steve: Well, the idea is you opt into using it, which saves you a $10 a month surcharge.

Leo: Oh, I see. Oh, good. I don't have to pay if I let you steal my stuff. That's good.

Steve: Exactly.

Leo: Good deal.

Steve: Exactly. And they're looking at your searches. They're looking at, well, basically, it is ISP-installed and sanctioned tracking, because ISPs want a piece of the action. And the idea is, you'll remember how nasty this is, this is modifying web pages which you are downloading, where the ISP is now inserting ads into web pages that were not delivered from the web server you're visiting.

Leo: Well, and I'd be annoyed if I were that web page.

Steve: Exactly.

Leo: If my ads were getting replaced. So I guess, I'm trying to think of how the spin would go. So your Internet service provider, you'll get a message from them saying, great news, we have a new technology that will customize the ads you see so you only see ads you're interested in. Now, if you don't want this, you can pay $10 not to get it. No, no, they won't do that. Oh, if you let us turn this on, we'll give you a $10 discount. That's how they'll do it. All right.

Steve: Yes. After raising your rates $10.

Leo: Yeah, first they have to raise your rates. Is this only in the U.K. right now, or is this going to be...

Steve: No, this is coming to an ISP near you, apparently.

Leo: How nice.

Steve: Now, the good news is, once again, SSL blocks it. HTTPS.

Leo: Can't do it.

Steve: It creates a secure tunnel from your machine to the server. And there's nothing your ISP can do to spy on you, as long as you haven't accepted a certificate from them. If that starts to happen...

Leo: They can intercept. They're the man in the middle then.

Steve: Then we've got a whole 'nother can of worms.

Leo: But nobody's going to know to do that. Comcast wants you to accept this certificate. Okay.

Steve: Exactly. That'll be really - that'll be the day.

Leo: So it's not illegal - I know there was some investigation going on when this first came around about a year ago.

Steve: Well, what happened was BT had done it without their users' knowledge or permission. It was being done behind their backs, and it caused a huge outcry. And, I mean, lawsuits flew because people said, hey, you're spying on us. So the idea is, this time it'll be done in an opt-in fashion with end-user knowledge. And there are statistics saying, oh, 60 percent of the people who we asked said they would opt in if they didn't have to pay $10 a month more. It's like, yeah, well, I guess so.

Leo: So if you're on British Telecom...

Steve: Well, I don't think this is BT again.

Leo: They've learned. They've been burned.

Steve: We're still early on this. I will keep an eye on it and let our users know where this begins happening. But it was being done for a while in the U.S., and U.S. carriers dropped it quickly because they realized this was just going to cause them trouble.

Leo: Terrible.

Steve: I did want our iPad users to know that iOS 4.2.1 is now available for iPads, adding all those cool new features that the iPhone has had for many months.

Leo: Including printing, which I've been playing with, and that's pretty nice, to be able to print.

Steve: Yeah, for me, I was able to condense all of my many apps onto just two pages, down from I think five.

Leo: Huge, yeah. I had 10, yeah.

Steve: So being able to have folders and aggregate them is really nice. And then I did have a fun "SpinRite saved me" story from a Security Now! listener. And the subject was "SpinRite Saved Me." Harry Lindenfeld wrote: "Steve, let me start by saying I've been a fan of Security Now! with you and Leo since Episode 1. Up until this past week I've never had a need for SpinRite since I have listened to you and Leo about backing up.

"I work at a courthouse here and inherited the security card access server and system for the courthouse from the county. Their solution to back up was CDs on a CD writer. The database was over 500 people, and the number of CDs required was getting out of hand and no longer made sense. We had a computer crash about two years ago, and I gave the county the backup CDs at that point. They were unable to restore the database, saying that some of the CDs were corrupt. So we had to manually reenter the complete database, which took days of painstaking work. I promised that would never happen again.

"So I purchased Norton Ghost and another hard drive and ran Ghost. Then I would do just weekly backups of the data files. Lo and behold, the main hard drive failed on the server, and I rebooted the system off the ghosted hard drive and was up and running in minutes. Two days later, that hard drive started to have issues, and I had not purchased yet another hard drive to replace the dead one on the main system. And the county still wants to use CDs.

"I was in fear of losing everything. So I purchased SpinRite for myself, because the county wouldn't buy it, and immediately started it in Level 2 mode. That was at 10:00 a.m., and it ran for just over four hours. By 3:00 p.m. that same day, I booted the system back up, and everything was back to normal. But here's the best part. The original hard drive that failed and wouldn't boot up, after running SpinRite on it, it booted up normally and was faster than before. SpinRite to the rescue once again. I have once again Ghosted the main hard drive and also have a second Ghosted hard drive offsite in a secure storage, just in case.

"Thanks again, Steve, for all your hard work. You've saved me days, possibly weeks, of data recovery. P.S.: SpinRite is now included in my bag of PC tools. Harry."

Leo: As it should be. Absolutely. All right. We're going to talk a little bit about DNS spoofing and how, well, we talked about how it worked in the past, but this is a name server spoofability test that you've got here that should be very interesting. Is this a new app from you?

Steve: Yes.

Leo: You're just cranking them out these days. Yes [laughing]. All right, Steve. I think we are ready to talk a little about spoofability. You want to recap a little bit the story behind all this?

Steve: Oh, of course. So here's the problem. When a user types a domain name into their machine, into their client, their computer, it needs, as we know, to look up the IP address of that domain if it isn't already known to the computer. So the computer sends a DNS query to the DNS servers that it's registered to use. Those servers either themselves then go about looking up the answer, or in many cases hand that off to, like, a master big-iron ISP server that does the work. But one way or another, some server gets the job of looking up the IP address.

So DNS was designed many years ago, at the beginning of the Internet, when, as we've often said, security wasn't even on the map. It wasn't even a consideration. Literally. I mean, it's hard to imagine that now. But it was absolutely the case that security wasn't even a consideration. Remember that Netscape introduced SSL on their later browsers. There was no way to even exchange information securely with a web browser. Truly, security wasn't considered in the beginning.

So a DNS query has a 16-bit query ID which was used by the server to identify the responses. That is to say, it would have a counter which would increment the 16-bit value, and it would send a query off to another DNS server asking it if it knew the IP address for this domain. And when the response came back, it would use this 16-bit query ID to verify that it was the proper response, to sort of segregate all of the outstanding queries with the returning replies.

What bad guys figured out was that, if the queries were always being sent out from the same port, and if the ID was just being incremented linearly, as they were originally, then it would be possible to spoof the reply coming back from a remote server. So the DNS server that's asking the question would generate its query. But because the query was predictable, because the port it was coming from was predictable, and the ID was predictable, it would be possible to beat the remote DNS server's reply with a spoofed one. So what that would do is it would be accepted, because it would be what the querying DNS server was expecting. It would be accepted as the truth. And that would poison the cache, poison the cache memory, the DNS memory of that DNS server, with the wrong IP.

So, literally, I mean, the way this would happen is a user could be going to Amazon.com, and the DNS server would have the wrong IP, deliberately, maliciously have the wrong IP for Amazon.com. It just wouldn't be correct. That would then take their browser to some malicious site masquerading as Amazon.com, and the site could do whatever it wanted.

So DNS spoofing is a very potent and powerful attack which the industry needs to prevent. So what Kaminsky suggested was, first of all, that the query IDs not be linear, that they be generated in a pattern which is extremely random, and the same thing for the port. The ports that the queries come from can be any of almost 65,536. 65,536 is 2^16, the number of combinations of 16 bits. And the query ID can be the same thing. It's 16 bits. So together that makes up a 32-bit source of randomness. As long as the DNS server is randomly choosing ports to issue its query and query IDs, both of those at random, then there's no way for an attacker to be able to predict a given query's port and ID; and it dramatically, to the point of it no longer being practical, dramatically lowers the spoofability of that particular DNS server.

So what I decided at that time, the summer of 2008, was that it was a perfect free service for GRC to offer, much as we've always offered the ShieldsUP!, test your ports, port-probing facility, which also was something that inherently had to be done from the outside in toward the user because that's of course the way attacks came at people. Similarly, what I realized was I could have GRC pretend to be a name server, and have the user's DNS server ask GRC for an IP. And I would then analyze the port and the query ID of the queries coming from the user's server.

Leo: Very clever.

Steve: Well, it gets way better. The first question I had to ask myself was, okay, I need a bunch of queries from the user's DNS servers. How do I generate those? How do I get - because if I give it a domain that I have control of, that it's never seen before, it won't be in its cache. So it'll have to ask for the IP. So I thought, okay, but I don't want just one. I need inherently, to do a good statistical analysis, I need hundreds, if not thousands, of queries in a very rapid order.

So I came up with a really cool solution. You simply go to GRC.com/dns, and that will bounce you to the Spoofability Test page. There's a button at the bottom. The button basically just takes you to another page. In order to display that page, which is the testing page, the web server tries to display, tells your browser to display the image for a really funky named, little tiny GIF image. It's 13 random characters. Actually, they're pseudorandom, and they never occur twice because they're based on a 64-bit counter back at GRC. So that way we know that the domain name that begins with those 13 characters has never occurred before. Then it's .dns.grc.com. So that's sort of a pseudo domain where this GIF image on the page that we're trying to display is located.

So that, of course, the page comes to your browser, and your browser says, oh, I need to display this GIF image in order to show the user this page. So it says, wow, look at that domain, I never saw that before. 13 random characters .dns.grc.com. So your computer asks your DNS server to get the IP of that domain where that GIF image is going to be served from. Your DNS server says, hmm, funky-looking 13 characters .dns.grc.com. So it's never seen the dns.grc.com in the same way that, for example, www.grc.com is a subdomain of GRC.com; dns.grc.com is a different subdomain.

So it asks GRC's name servers for the IP address of dns.grc.com, and it gets a special IP for a pseudo name server that I wrote. It says, okay, now I've got the IP of dns.grc.com. What's the IP of this funky 13 characters .dns.grc.com? It is a subdomain of that subdomain. Well, there's a record, there's a type of reply in DNS called a CNAME, a canonical name, the idea being that, if you ask for a domain name, that could be an alias for something else. So when you ask the GRC.com pseudo DNS server for this funky 13 characters .dns.grc.com IP, this pseudo DNS server that I wrote returns an ungodly alias, that is, the actual canonical name. It is a.a.a.a.a.a.a.a.a.a.a - 43 a-dot subdomains, then that same 13 characters .dns.grc.com.

So think about that, Leo. It's as if you had a 43-deep hierarchy of domain names. And what this forces is, this forces individual DNS queries to enumerate each of the IP name servers for that entire hierarchy of sort of fake DNS. So the user's DNS server receives this canonical name which is incredible, I mean, it's never seen anything like it before. But it's valid. And it says, oh, my goodness, okay, let's see. I need to get the name server for the a.13characters.dns.grc.com. And it asks that name server for the a.a. And then it asks that name server for the a.a.a, and so forth, basically exploring all the way out to the end of this insanely long, deep domain name where - I used a.a.a. because that's a single character, which is the fewest you can have between dots.

And so what this does is this forces a flurry of queries in very short order between the user's DNS server and my pseudo server, which is the name server for all of those subdomains, down that path, allowing me to collect in a database for this user, for that particular querying name server, all of the ports that the queries come from and all of the IDs that the queries have.

And so on the fly I build databases for all the DNS servers heard from. This is also deliberately slowed a little bit because, I mean, we're doing a lot of work here with this going back and forth. but we also don't want to respond too quickly because one of the other things that happens is, because I want to discover all the name servers that might be brought to bear, not just the particular name server this time, but I want to do a comprehensive discovery of name servers. So the system also throttles its replies so that at least some number of seconds, typically five seconds, will be required for this entire resolution process.

What happens is the client that asks the question of its DNS server gets impatient. After a second, it asks it again, thinking that maybe its query got dropped. And then, if it still doesn't hear back after another second, it says, well, maybe that DNS server has died. So that's where the secondary name server, or more name servers, because some users - there's no practical limit to how many name servers you can configure on your computer.

So what Windows and Mac and UNIX and normal protocol-obeying clients do is, if they haven't heard back after two queries a second apart, they then ask all the name servers that are registered for them to use the same question. So suddenly more name servers are being asked for this domain. They start querying GRC. I collect all of this stuff together. Because this 13 characters is changing every time, I track which ones are associated with which users trying to run the spoofability test to aggregate all the data. And essentially that approach allows me to discover all the name servers which might generate public queries out on the Internet for a given user, based on their current DNS configuration. And I collect this rich database of associated query IDs and query ports, each being 16 bits, and assemble that into a report.

So the processing done, once this has all happened - oh, and this actually happens multiple times. What we discovered during the testing of this is that the longer you waited, the more name servers you found. So this test continues. It shows you little progress dots as you're performing the test, as it moves along, so you can see what's going on. Shows you how many name servers have been discovered so far, accumulates all this data, and then grafts them in paired scatter charts.

I display the results visually because, good as software can be, nothing is better than the human eye for picking out patterns. And so you can instantly see whether this just looks like true scattered queries with no pattern, or whether there are any kinds of, like, zones of queries, or vertical or horizontal or diagonal lines. It turns out that we're very good at seeing these results. And in fact a ways down the page, where lower down I'm explaining all of this, I give a link to a gallery of sample DNS name server scatter charts that are really bad, which our users, which users during testing of this found their own DNS servers were producing. And, I mean, it's a source for some worry.

And then in addition to that, in addition to these scatter charts, since there are anomalies that the human eye might not see - for example, say that the query IDs were always odd. Well, you couldn't see that in a scatter chart. But you could notice, that is, software could notice that the least significant bit of the query ID was always zero, meaning that it would be - wait, no, it was always one, meaning that it would be an odd number.

So I then also create some charts showing the bit predictability of each of the 16 bits of the query ID and the source port, and indicate, if the predictability is not near zero, then some person could profile the server, just as I have, a malicious attacker could profile the server as I have and notice that there are bits that are stuck in some cases, or highly predictable. And then what that does is that lowers the effective entropy, the effective randomness of these queries, making that server more spoofable.

Then, additionally, I do some statistical analysis to determine the maximum entropy, the lost entropy, a bias in the direction of the bits, and basically fully characterize and profile the name servers which are producing public DNS queries on behalf of the user out onto the Internet, and summarize it, and actually just tell you flat out if your antispoofing safety is very good, good, moderate, not so good, and so forth down the scale. And then the balance of the page explains everything that happened above. And it turned out very nice.

Leo: Yeah, I have to say, I'm looking at the scatter charts. If you go to GRC.com/dns, you can read all about this. There's all the details and so forth. And then there's just a little button at the bottom that says "Initiate Test."

Steve: Now, one glitch happened as we were developing this, Leo.

Leo: Yes?

Steve: We crashed some people's routers.

Leo: Oh. You should have told me that before I pushed the button.

Steve: Yeah. We crashed some people's routers. And people were like, wait a minute, I tried to run your test, Steve. This was organized in our newsgroups during the development of this. And it was like, okay, wait a minute. I wasn't getting myself crashed. Other people weren't. I was only issuing valid DNS queries. It turns out that some firmware in some routers is buggy.

Leo: Well, in fact you'd probably want to know that; right?

Steve: Exactly. Because what we know...

Leo: By the way, I'm killing your Skype right now. I'm sorry. Does this send a lot of traffic?

Steve: Oh, my goodness, yes.

Leo: Okay, sorry about that. Just keep talking, it's okay. I probably should stop the...

Steve: No, I'd love to see what you see in terms of your own results.

Leo: Yeah, well, now you've got me going. I'm not sure which - I don't know whose system we're using right now. We have so many different ISPs in The Cottage.

Steve: Do you have little dots? You're looking at little black dots move across?

Leo: Little dots are going across, yeah. We've got three query rounds so far. Found four servers in the first round, 649 queries received. No servers in the second or third round.

Steve: Okay, good. So it looks like you found them all in the first round. What I do is I keep doing additional rounds because sometimes there are reluctant DNS servers that kind of, like, finally appear. So we get four rounds of zeroes. And then I've decided, without finding any new DNS servers, then I decide that we've found them all.

Leo: I suspect I'm OpenDNS on this one. And of course they patched BIND right away. In fact, I don't think they were ever vulnerable. As I remember.

Steve: You're right, I think that they were always very safe.

Leo: Well, we don't have time for me to go through all of this. But I'll keep - oh, wait a minute, it's done. Oh, it's done. Oh, that was fast. As soon as I said we don't have time, it finished. Okay. 809 queries from server at - oh, I guess we're not, 68.87.76.180. That's not OpenDNS. Antispoofing safety good. It is Comcast's server.

Steve: Okay.

Leo: And the distribution looks completely random, all over the place. So query source port analysis looks excellent. Transaction ID analysis, excellent. So I have no idea what any of this means. But it looks pretty good.

Steve: But that's only one server. Scroll down to the next one.

Leo: Oh, my God. And here's another one. Two of those were in the San Francisco - that was San Jose, San Francisco. Utah, Salt Lake City. This is all Comcast. And they all look good.

Steve: That's very good. Well, so what this did was this found DNS servers all over the place that have the ability to serve your DNS queries because this page was crafted to generate a huge number of DNS queries and then analyze the results.

Leo: The only reason I think I got "good" instead of "excellent" is lost entropy. There's a little lost entropy on all of them.

Steve: Okay.

Leo: I don't know what that means.

Steve: It would be a function of, like, maybe the bit predictability on either side are not, like, really down near zero?

Leo: It's pretty low - .18, .2, .24. That's pretty low; right?

Steve: I don't think you have anything to worry about, obviously.

Leo: It's funny, though, that it's just not binary, that it's not like, oh, it's okay or it's not okay.

Steve: Well, because it isn't. In fact, you could have, for example, some servers, it happens that when I brought this page up for myself, I didn't have any source ports below, it looks like maybe below 4,000. So it was only from 4,000 up to 65,536, which that's still a large range, but it means that none of the source ports were from port 1 up to 4,000, which eliminates a huge clump of possibilities, meaning that an attacker wouldn't bother guessing any of those. So it really is a variable thing. It's not just go or no go. It really is how variable is it.

And in fact, Leo, now that you've got a page there, you scroll down to the bit predictability chart description, right above it you'll see the gallery of sample DNS name server scatter charts. If you click that, it'll give you an idea of what some users who are listening to this podcast will see because, even now, DNS servers have not been fixed because there's been no pressure put on them to get these things fixed. So my hope is, one of the reasons I did this, was this would give end-users a tool to say to their ISPs, hey, I'm using spoofable DNS servers. Fix these.

Leo: Just looking at the scatter chart tells you right away. If it's totally random-looking, that's good. But the less random, the worse it is.

Steve: Yes, sometimes you'll see diagonal lines, meaning that there are counters running instead of something being pseudorandom.

Leo: So this is how that whole spoofing occurred, because it wasn't a random assignment.

Steve: Yup.

Leo: So the more random, I mean, if it looks random, it probably is. And if you see any patterns there, that's not good.

Steve: Right.

Leo: Look at that diagonal line. That's wild.

Steve: Yeah, isn't that? That just had a simple counter, just like...

Leo: Terrible.

Steve: It couldn't be any worse.

Leo: Terrible.

Steve: It could be a straight line, which would mean it was always the same thing. That would be worse.

Leo: That's pretty predictable.

Steve: But on the issue of router crashing, as we know, exploits start out as things that cause crashes. And then the bad guys analyze the crash and figure out a way to execute code instead.

Leo: So if it crashes your router, that's not good.

Steve: Yes. What we've discovered is that it is possible to crash some consumer routers by returning to them a DNS query that is legal, but the router doesn't like. So what happened was, we found two different sources of trouble. One was the length of the query, which is why I trimmed it to 43, so as not to crash people's routers. And also, when we finally gave the answer to the final question, when they went all the way out, the a.a.a.a.a, all the way out to the end, if I gave them the final IP, that would crash people's routers. So the normal test, the Spoofability Test, deliberately doesn't crash anyone's router.

But I created a second test called, not surprisingly, the Router Crash Test. And the Router Crash Test is also there at the same location. You can find a link to it at the bottom of all of the DNS spoofability pages called Router Crash Test. When you click that, what it does is it issues the most aggressive, still legal, but most aggressive queries, only a few of them, and a huge number of Belkin routers just go belly-up immediately. They just die.

Leo: Oh, that's good to know.

Steve: And a number of others. In fact, what we have is there's a feedback page for people to let me know if their router does or does not crash, and is or is not listed among those that do. And so I'm maintaining on the site, on this page, a list of all the routers by model number and firmware version which are known to be crashable by this Router Crash Test. And again, my hope is that this will put pressure on the router vendors to fix their firmware because nobody wants a router that my website is able to crash. Because if I can crash it, and I'm doing so non-maliciously, somebody else could crash it maliciously, and they may be able to do more than crash it. We don't know. But they may be able to execute code, which would be an external code execution on your router from the Internet. And that's about as bad as it gets.

Leo: Wow. Very interesting. GRC.com/dns, that's where to go to do this. I've stopped now, so I'll stop screwing up your Skype. Great stuff, as usual. If you want to know more about this stuff, GRC.com/dns, in great detail on how it works and also how the spoofing attack works. If you want 16KB versions of the show, you can get them at the same spot. Of course full transcriptions are available, too. Steve pays for those - thank you, Steve - at GRC.com. While you're there, buy yourself a copy of the fabulous SpinRite. Everybody ought to have a copy. If you've got a hard drive, you need SpinRite. And we will be back next week. Have a great Thanksgiving, Steve.

Steve: Thanks, Leo. We'll do a Q&A, talk about maybe the Benchmark and the Spoofability Test, follow up any questions that our users have. And we'll go from there.

Leo: Good. Just if you have a question, GRC.com/feedback is the place to go. We'll see you next time, Steve. Have a great Thanksgiving.

Steve: Thanks. You, too, Leo.