SECURITY NOW!

Transcript of Episode #155

BailiWicked Domain Attack

Description: Steve and Leo discuss the deeply technical and functional aspects of DNS, with a view toward explaining exactly how the recently discovered new DNS cache poisoning attacks are able to cause users' browsers to be undetectably redirected to malicious phishing sites.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-155.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-155-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 155 for July 31, 2008: How DNS Works. This show is brought to you by listeners like you and your contributions. We couldn't do it without you. Thanks so much.

Time for Security Now!, Episode 155. We're talking about security, protecting yourself on the Internet, learning how the Internet works, a lot of great topics with a guy who certainly knows this subject inside and out, Mr. Steve Gibson, the guy who coined the term "spyware" first, first guy to discover spyware, has made so many great security utilities on his website, GRC.com. And of course best known, well, I don't know what best known is, Steve, but certainly well known for SpinRite, his hard drive maintenance utility. But also you got well known, I think I first became aware of you with your InfoWorld column.

Steve Gibson: Yep, I think that was probably early on. Actually it was fun, it was for the promotion of SpinRite that I approached InfoWorld. And I said, hey, would you be willing to trade an ad for a column? And they were like, uh, who are you? I somehow talked them into it. And so we just did a trade-out deal. I didn't need any other compensation, but just having a chunk of space to let the world know about SpinRite was my goal. And of course the column ended up really taking off. I wrote it every week for eight years.

Leo: I loved it. I was so sad when you quit.

Steve: Well, it just, you know, I did it for eight years, and I pretty much said all I had to say. And I was getting busy with other stuff.

Leo: Sure, sure.

Steve: And so it was like - and frankly, I remember, too, as you probably remember, the magazine really began to change, not for the better.

Leo: Well, it became a business magazine instead of...

Steve: The techie magazine.

Leo: It was the journal of record for Silicon Valley. Between you and Dvorak, those were...

Steve: And Cringely.

Leo: And Cringely. You had to read that magazine every week. And I did, and I loved it. But then it became really more like a business magazine, and I became much less interested in it. And of course it faded away as a result.

Steve: Uh-huh.

Leo: What was the name of your column?

Steve: It was TechTalk.

Leo: TechTalk.

Steve: Yeah, I originally named it Behind the Screens.

Leo: I like that.

Steve: I did, too, except that it turns out that CompuServe made a trademark claim to having Behind the Screens. And it's like, okay, fine. So I got a call from InfoWorld. They said, okay, Steve, you've got to rename your column. And it was like, uh, well, okay, TechTalk.

Leo: Yeah, that's a good one.

Steve: It flew forward as TechTalk for eight years, and it was a lot of fun doing it.

Leo: What are we going to talk about - speaking of tech talk - what are we going to talk about today, Steve?

Steve: Well, our plans have changed a little bit from the last two weeks we've been talking about what we're going to talk about today because essentially what is probably all of the information that we were going to embargo for two weeks until after Dan Kaminsky had officially released it during his planned-for talk on August 6 during the Black Hat conference in Las Vegas, all of the information has made it out into the Internet. So it's no longer necessary to embargo it.

And we're going to talk about, first of all, how DNS works to create a real good foundation for understanding how it is that it is possible to exploit it, essentially what it is that Kaminsky realized early this year. It isn't a bug. It wasn't a defect. That is, it wasn't, like, coded wrong, which is why it affected Microsoft DNS and the main BIND issue. DNS essentially Internet-wide turns out to be more readily exploitable than was believed before. DNS spoofing is not new, but this is basically a new, really more potent type of attack. So we're going to really explain - and I've never done it in our nearly three years of the podcast, we've never really focused on DNS. So I want to lay down a nice foundation of understanding, which is why people need to get their propeller-head caps going. And then we'll talk about, because we'll understand how DNS works, we'll be able to talk about what it is that makes it vulnerable and what the vulnerability is.

Leo: Yeah. I'm surprised we've never covered how DNS works. That seems like a - because early on in the show - and I highly recommend people go back and listen. All of the episodes are available from Steve's site, GRC.com, or on TWiT.tv. We covered really a lot of the fundamental technologies of the Internet.

Steve: Yeah. I referred to name servers and DNS spoofing and man-in-the-middle attacks. And we've sort of - we've talked about it tangentially, or maybe it's circumferentially, I'm not really sure. But radially. But we've never really just said, okay, let's nail this down. And interestingly, it is not a very well understood technology. We all sort of - people know, oh, well, I've got DNS servers somewhere. I point my computer at them. And of course we've talked about OpenDNS on several occasions when DNS issues arose. But we've never really explained in detail how it works. And there are - as always, it's in the details that this attack functions. So we're going to explain it all the way, completely.

Leo: So what's going on in security news? Besides this Dan Kaminsky story has been just dominating the headlines for three weeks now.

Steve: Well, yes, it has. And essentially the news there, because there's a little bit of news, I alluded to before. And that is that essentially, as is so often the case, when you know that there's a problem, you're much more likely to go and see if you can figure it out than if you don't know there's a problem. And when I started thinking about, okay, if I were to test for this problem for visitors who come to GRC, how would I induce their name server to generate a bunch of queries so that I could see what it was doing in order to measure its randomness? And it turns out that that's one of the ways, that's one of the components of this attack. So what happened was, over the course of the last couple weeks, unfortunately for maybe for some people who get caught by this, which essentially is going to be like a major phishing trap, unfortunately a lot of people in the industry began speculating. And some people started guessing and posted their guesses on blogs. And other people said, no, no, it won't work because of that, but this would. And what happened was basically...

Leo: People figured it out.

Steve: Figured it out. It ended up being figured out. And then finally one security researcher made a guess, and then somebody who was actually in the loop, who had been told directly by Dan what the problem was, confirmed it.

Leo: So Dan wasn't going to say anything until Black Hat, which is next week, right, or the week after.

Steve: Yeah. It's next week. And then our original plan was to have him on the following week, and just as our guest, to hear it from him. But basically, unfortunately, his talk has probably been preempted. Now, it's worth noting, though...

Leo: Oh, he'll still talk, and still give it the full treatment, I'm sure.

Steve: Oh, well, yes. But it's worth noting that we don't know that we've figured out, we the industry have figured it out. There could still be something else. What is known is that something new has been found and is being exploited. And there are now exploit tools for this.

Leo: Right.

Steve: Anyway, so there's that. But I did want to mention two things. And this, of course, is the big security news. And blessedly, there wasn't much else that went wrong in the last week...

Leo: Hallelujah, yeah.

Steve: ...since we last talked. Yeah. Except that another problem was discovered in Mozilla-derived browsers. There's a memory corruption bug which was already fixed by the most recent release. But it's a more serious problem than the reason we went to, in the case of Firefox 3, to v3.0.1, and in the case of Firefox 2 to 2.0.0.16. So I wanted to reinforce in our listeners' minds, I said this last week because there were other things that those were fixing. But a more serious problem came out. So I just wanted to make sure that our listeners who are using Firefox had brought themselves up to date because there are some other things that ended up being fixed sort of in the process that have since been found to be wrong with the down version versions.

Leo: Excellent, okay.

Steve: And my only other news is I got a really interesting SpinRite story from one of our listeners, Alain Donais. He said, "I bought SpinRite recently, and talk about great timing." He said, "Right after I bought it, my parents called me because their computer won't boot. And since it's still under warranty they brought it to the store's technician, where he told them that the hard drive was done," as Alain says.

Leo: Done.

Steve: It's done. Oh, it's done.

Leo: It's cooked.

Steve: And it had to be replaced. "This was a big problem as they had no backup of their photos and other important data. So I told them to bring the computer over, and I ran SpinRite. After several hours the computer was working as new. The tech at the store was extremely surprised to know that I had fixed my parents' hard drive. The next day, and I kid you not, when I get home from work, my co-tenant informs me that his computer's power supply died, and after replacing it the computer won't boot. So I take out SpinRite, and after several hours another hard drive fixed. And finally, recently, my brother had to replace his computer..."

Leo: No. Wow.

Steve: "...because his very old computer was dying on him. And after putting his old hard drive in the new computer to transfer the data, the drive is unreadable. And once more, SpinRite to the rescue. So as you can easily figure out by now, this was one of the best investments of my money ever. Thanks, Steve, for writing such a useful and efficient software." Signed, Alain.

Leo: That's really neat. I actually have a SpinRite-didn't-work-for-me story, as you know. But it wasn't - one thing I think we - I try to make this clear is that SpinRite is for a particular category of problems. But there are other kinds of hard drive problems. For instance, if it's a file system problem, your partition table got mucked up, then SpinRite won't help. It helps with bad sectors, right, I mean, that's really...

Steve: Well, it also helps, for example, many times we've read testimonials where people say that their machine is in an infinite reboot loop.

Leo: Right.

Steve: Well, it's probably, I mean, it's a sector-level problem, but so the sector is being read incorrectly or not at all, and so that's causing SpinRite to keep from booting. So that is a file - it's in the file system, so it's a file system problem. But in your case, Leo, I mean, it just stopped being a drive. It became a doorstop.

Leo: Exactly. We use a very cool device called the TriCaster for the video, the TWiT Live video stream. And it allows me - it's basically a video control room in an XP box, the size of a Shuttle PC. And, you know, it allows me to switch cameras to put CGs up, record video, playback video. And it's really great. But it's a computer, and the hard drive - we're in the middle of recording, and blue screen. And so I tried to reboot it, and it said no operating system. And so I knew immediately it's a hard drive; right? And of course I knew immediately what to do. I pulled it out, put it - because this doesn't have, which is a little frustrating, doesn't have optical drives. There's no room for them. So I had to put it in another machine so I could boot into SpinRite. An interesting thing happened. And I was curious as to what your take on this was. The BIOS says, yeah, there's a drive there, gives me the correct ID number. But the BIOS says 0.0GB drive.

Steve: Yeah, that's an indication that the drive isn't even online. I mean, it's responding to some commands. But there's a low-level fundamental command that says tell me about yourself. And if the drive is unable to do that, as you suspect, it's probably the circuit board on the drive, not the actual magnetic media on the drive.

Leo: Yeah, because I don't hear any noise coming out of the drive. It's spinning up fine, you know, it sounds like it's acting normally. You know, I'm used to bad sounds, so like access sounds [vocalizing], none of that.

Steve: Can you hear it, like, loading the heads? Like, is there like some fluttery sound in the beginning?

Leo: Yeah, at the beginning, [vocalizing], yeah. And so it spins up. I hear the heads, well, actually, you know, that's a good question.

Steve: Because it might just spin up, and the heads never even...

Leo: I think it's just spinning up. The heads never start moving, I think. I'll have to listen.

Steve: It sounds to me like the little motherboard or the daughterboard, some child sibling board on the bottom of the drive is just, well, and you've only had it for, what, maybe a month?

Leo: Yeah, couple of months. And it just failed.

Steve: So it's probably classic instant mortality, where one of the electronic components on the board just gave up. It passed all the QA, it got through manufacturing, it made it through the TriCaster folks. But there was still some fundamental weakness that finally manifested itself. And that's an example of something that, I mean, nothing can fix except somebody like a DriveSavers, who would have an identical drive and would be able to, like, pull the board from it and then swap it onto yours.

Leo: Yeah, that's what they do, they actually put a new circuit board on. That would be the first thing. If I had a circuit board, if I had a matching drive and I had the circuit board, I'd try that immediately because that's sure what it looks like. And of course if the machine can't access the drive, SpinRite can't, either. SpinRite's not magic. If there's no physical drive to see, it can't do anything. So I immediately booted SpinRite, and it didn't see a drive.

Steve: Yeah. And the good news is that we all know how much more reliable solid state stuff is than magnetic state stuff. So I don't think I've ever, maybe once or twice in 20 years I've seen an instance where a hard drive's daughterboard died. Typically it's not that. It's sector problems, and of course that's what SpinRite is made to fix. So normally it can help.

Leo: So the good news is NewTek, which makes the TriCaster, immediately sent a new drive because we're just going to swap it out. And this, you know, and people got mad at me. They said, Leo, you always say back up. Well, I was backing up. And we didn't lose any data. We backed up all our videos except for the one video we were making while we - when the drive died.

Steve: That was actually in, yeah, exactly, in there, in the can.

Leo: And I might bring it to DriveSavers just to get that video back, or just for fun to see if they can, see what was really wrong. But what I didn't do, and I should have done, is made an exact copy of that drive so we could just swap another drive in. And that's what I will do this time. I'll just...

Steve: Well, and you probably thought you had a little more time before it became critical, too. I mean, this did catch you off guard.

Leo: It did.

Steve: It was still brand new.

Leo: Yeah, and we had planned to. I had told Colleen, by the way, let's make an image of this drive and keep it safe so that we can restore if the drive dies. In this case I don't think I want an image. Actually your consultation on this would be also useful. I don't think I want an image because I don't want to restore an image. What if - if this happens again, the drive's dead, what I really want, I think, is to be able to pull that drive out and put an identical one in and just get going again immediately; right?

Steve: Right. So you want to make a clone.

Leo: A clone.

Steve: An exact, bit-for-bit duplicate, you know, partition table, partition sectors. And being that this is like a proprietary drive from inside of a device, they might very well have some anti-copy-protection stuff. I don't...

Leo: Exactly. I don't know what's on it. Yeah, we don't know. So that's why I think a bit-for-bit copy is what I want to do.

Steve: Yup.

Leo: Yeah, okay. And I'm going to try - there are various applications that'll do this. My instinct is to try one of these free ones that all the drive manufacturers - I have an Hitachi that I'm going to make a duplicate on. And I think all the drive manufacturers offer these bit copiers, you know, like the Max Tools and stuff, so that when you buy a new drive you can just take the old drive and make an exact copy and then put the new drive in.

Steve: Yeah, they're wanting to promote your migration upward to an ever-bigger drive.

Leo: Precisely. I'll have to see if Hitachi - is Hitachi an okay - I got a deal on it.

Steve: As a matter of fact, I was talking about the big monster machine that I recently built, the quad core with the - I think I've got five 1TB drives in a RAID. I chose Hitachi drives.

Leo: Oh, really.

Steve: I think they are absolutely among the best.

Leo: Oh, excellent.

Steve: I feel very good about Hitachi.

Leo: Patrick Norton tweeted on Twitter than Newegg was selling 750GB drives for $119 plus an instant $20 rebate for a hundred bucks, for 750GB. So I bought two. And there was a choice between Seagate and Hitachi. And I love Seagates; I've used Seagates for years. But the Hitachi had a bigger onboard cache and, it looked like, better sustained throughput. And I thought, well, I'll take a chance. So that's good to know that it wasn't a mistake.

Steve: Yeah. And the cache is important also for, you know, you're doing heavy bandwidth media stuff.

Leo: Exactly, exactly. Yeah, this is a great, I mean, these are amazing how cheap these drives are.

Steve: Incredible.

Leo: But because they're so big, I'm with you, I'm a little nervous. I'm going to make a clone. I'm going to have a couple of clones just in case.

Steve: Here come the clones.

Leo: All right, I'm prepared now. I've got, you know, I think a lot of times when you talk I like to jot down notes. So I've got my notepad in front of me.

Steve: Okay.

Leo: And we're going to talk about DNS.

Steve: Okay. So back before DNS, the way the Internet - whoa, we're having an earthquake. Oh, a big one.

Leo: Oh, he's shaking. Oh, you're really shaking.

Steve: Goodness.

Leo: Hold on, Steve. Hold on. I'm watching the books behind you. They look like they're okay. But you're getting quite a temblor there.

Steve: Wow.

Leo: That was a big shaker.

Steve: That was an earthquake.

Leo: Holy cow. Wow. Steve's in Irvine, California.

Steve: Yes. On the San Andreas fault line.

Leo: Are you?

Steve: No, no, no. But it's, wow, it's still going on.

Leo: Wow.

Steve: Somewhere there was a big earthquake, probably not that far from here. But this was more roll-y than jerky. And the closer you get, the more abrupt and sharp it feels.

Leo: That's right. So you know this is maybe a little bit distant.

Steve: Wow. That was...

Leo: You know, I've only had one other live earthquake. We had it on the...

Steve: Sorry, folks.

Leo: No, don't apologize. That's amazing. We were doing The Screensavers, and I'm talking along and oblivious. And Kate says, "Did you feel that?" And I said, "What?" "We just had an earthquake." And I had no idea. But this one I could tell. We could hear it. We could hear it rattling.

Steve: Yeah, well, all sorts of stuff fell over all over the place, too. I've got to do a little cleaning up now.

Leo: Do you want to pause?

Steve: Oh, no, no, I'm fine. Yeah, that was - I'm in California. We're used to earthquakes.

Leo: Holy cow. That's a first. That is a first. By the way, up here in Northern California - I'm about 500 miles away from Steve - we didn't feel a thing. So it is something probably centered down there in Southern California.

Steve: Probably out in the desert is normally where they are. So I imagine you'll have some people in the blog checking the earthquake log, and we'll know before our podcast is done how big it was and where it was, so.

Leo: Pasadena had some. Dogs are going crazy in Hesperia. So, yeah, Apple Valley five seconds after Steve said something; felt it in San Diego six seconds later. It was probably pretty close to where you are, actually.

Steve: Yeah, it was - it's the biggest earthquake we've had in maybe five years.

Leo: Big shake in Fontana. Yeah. I could see it.

Steve: Things have been quiet actually. So I'm glad to have a little stress relief because you don't want to let them build up too much.

Leo: You're right, much better to have something like you just had.

Steve: Lots of little ones, yeah.

Leo: Yeah. Wow.

Steve: Okay.

Leo: Back to DNS.

Steve: So once upon a time there was actually a single hosts.txt file which maintained the machine names of everything on the Internet. It was one file. It was maintained by the original Internet authority, NIC. And they would maintain it, and they had it on an FTP server. And every...

Leo: No, wait a minute. You're kidding me.

Steve: I'm not kidding you.

Leo: So it was automatic? I mean, wasn't automatic, you would actually FTP the tables?

Steve: Yes. Whenever - periodically other people on the Internet would FTP the master, single master hosts.txt file to their machines, and that was the only way their machines had of converting names to IP addresses. And of course, needless to say, that became rather unwieldy after a while as changes began happening more often, as the single master hosts file for the Internet became extremely large.

Leo: So people were running locally the software that would do the lookup, and they would download the tables kind of at will.

Steve: Well, there wasn't even software that would do the lookup. It's like our hosts files that you and I talk about all the time? It's on their computer. And so their email, their web browser, well, even before, their gopher client, their IRC or their Usenet newsreader, those would look in the hosts file. Actually the hosts file is supported by the so-called "sockets layer" in UNIX. So there were low-level commands that were sort of an API in the networking stack, where you could say look up the address, the IP address of this machine. And all it would do is look in your hosts file. There was nothing like DNS.

Leo: So but that's interesting because that's really where the hosts file came from.

Steve: That's its origin, exactly.

Leo: I just want to - just let me break in. CNN is now showing pictures of a smoggy Los Angeles and saying there's an earthquake. The U.S. Geological Survey is reporting a 5.8.

Steve: Wow.

Leo: Which is a fairly serious earthquake. They're saying it's roughly the Chino Hills. But you're pretty close to it. And it is Greater Los Angeles. Now, given the size of that and the fact that it is an urban area, I imagine we will start hearing some damage reports soon. But just to give you an update on that.

Steve: Yeah, cool, neat. Okay.

Leo: Not so neat if you're in a building that fell over.

Steve: Yeah, not if you're near Chino Hills.

Leo: You know what's impressive is the Internet's rock solid. Skype was rock solid.

You were shaking, but everything else was fine. Video looked good. Few books fell off the shelf, but that's it.

Steve: Okay. So what happened was, as this hosts file, the Internet master hosts file, became unwieldy, the engineers decided, okay, we need something better. We need something scalable. We need something distributed. And they had a whole bunch of criteria. The idea was they wanted what they call a "hierarchical name space." And what that means is that you would not - machines would not be known just by a name because of, well, the problem of name collisions. It's like, for example, you can't get an eBay account under your name anymore because somebody else already has it. And so it's yourname3267295 or some nonsense. And so, which is annoying. On the other hand, you can have your name if it's in a subdomain. And so what they did, they created this notion of a hierarchical name space. And that's what this com and then eBay and then www, there's three levels of hierarchy - www.ebay.com. And the hierarchy is shown by the periods in a domain name.

So their idea was that there would be a set of servers which would know the IP addresses of sort of the next layer down. But those top-level servers, the so-called "root servers," they wouldn't know anything about, for example, eBay. They would just know about .com. And so the idea was that this would create a tree, essentially, where there's a root to the tree, the so-called "root servers," that are aware of the first level in the hierarchy. And then the second level in the hierarchy is known to that first level, and the third level is known to the second, and so forth. So that's where this whole something.something.something.something comes from. So what happens when I want to look up some domain that I've never been to before is in my own machine, Windows or Mac or Linux, we all have a local cache, that is, our machines will store the IP addresses of domains we've looked at since we've been powered on, or until they have expired.

Leo: That's independent of the hosts file. That's in memory.

Steve: Actually that's - it's in memory, but it's after a check for the hosts file.

Leo: Oh, interesting.

Steve: The hosts file is still in place.

Leo: That's the first thing that gets checked.

Steve: Yes. And what's cool about that, and we've talked about this in a couple different contexts in the past, is by putting things in the hosts file, you can preempt your computer from making a true query out onto the Internet's DNS system to look for something. Which for example is how you could, if you didn't want your computer ever going to DoubleClick.net, or you didn't want it ever going to certain bad sites, you know, you can download hosts files which essentially prevent your machine from going to a whole range of domains which you've decided, or the people who maintain this hosts file have decided, are just bad. You don't want your computer going there. You stick that list in the hosts file, and it preempts your machine from making a query.

So the other thing that it's important to understand is that all of the information in the DNS system has an expiration time, that is, it says I'm some information, and I'm good for six hours. And what that means is that as the information is obtained by DNS servers, they cache it. They store it, and they keep track of how old it is so that when a query comes for that information, if it has expired, the DNS server will be prompted to go update itself and get a fresh copy. And this is a wonderfully clever solution for the problem of a distributed database like this because there's - where we've got it, we've got the information spread out all over DNS name servers all over the Internet. It makes no sense if every single time someone makes a query we have to start up at the root and work our way back down to find the information. First of all, the root servers would just be overwhelmed. And we're getting then no benefit from this distributed hierarchy.

So I'm at my computer, and I put an address into my web browser. And first my computer looks to see if it's in the hosts file. If not, it looks to see if it's in my local cache, that is, does my computer already have the address because, for example, I put the same address in five seconds ago or five minutes ago, and it hasn't yet expired. So if my computer does have a copy, nothing happens except it just looks up within its own cache the IP address. If it's not in my computer, then my computer makes a query out to the DNS server that it's been configured to ask.

Now, that's for most customers, that's sort of an automatic operation. For example, in Windows, if you say "obtain IP address automatically," your machine gets an IP address, and it also receives the name of two name servers, that is, two DNS servers that have been approved by your ISP and are typically provided by your ISP, either actually by them or they've got a deal with somebody else that allows their customers to use some other DNS server. One way or another, your machine has the IP addresses of two DNS servers. And two is an important number also. It's just it's always been the case that, for the sake of redundancy, we don't want to rely on a single DNS server. So all domains have a minimum of two DNS servers, sort of just by administrative fiat. The original gods of DNS said there shall be two, at least. The idea being that, if there were one, and it were down, basically you're in trouble. You're not going to be able to figure out the name of something on the Internet. So for redundancy we've got - and just by convention there's always at least two.

So your machine sends a packet, a DNS request packet to the remote DNS server located typically at your ISP. Now, what's interesting is this uses the UDP protocol rather than the TCP protocol. We've talked about TCP and UDP in the past a number of times. But we'll discuss it here briefly because this is part of the problem with DNS, which is also part of the economy of DNS. The original DNS designer said, okay, a query is a tiny thing. It's just basically a what's the IP of www.GRC.com, I mean, a very short request. So it will fit in a single packet. So - I mean, with lots of room to spare. So - because a packet can be, like, 1,500 bytes, and this probably takes 100 bytes. So they decided it makes no sense to establish a TCP connection when that requires sending a SYN packet to the server, it responds with a SYN/ACK, we respond with an ACK, so that's the so-called TCP three-way handshake. Then we send our query, and then it sends back the answer. Then we say, okay, we're done, so we send a FIN packet, and it sends a FIN/ACK back to us to shut down both directions of the conversation. So that's a huge amount of packet traffic just to send 100 bytes.

And this is exactly what UDP was designed for. It is a connectionless protocol. It is the most popular connectionless protocol. And so essentially a single packet gets sent from my machine to one of the IPs that's configured for DNS on my machine. And it says - basically that packet is a query saying what is the address of GRC.com? So if the server already knows, it'll instantly send me its response. The reason it might already know is, first of all, I might have asked it, for example, on a different computer here in my network. So this computer didn't have that in its cache, in its local cache, but another computer might. So I go to a different computer here, ask the same question.

Well, the point is that my ISP's DNS server is serving DNS for a huge number of ISP's clients. So super-popular domains like Microsoft.com, like...

Leo: Google.com.

Steve: ...Google.com...

Leo: Or Amazon.com.

Steve: Yeah, exactly, Yahoo!. All of those are almost certainly already in its cache. So before it responds to me, it'll make sure, first of all, that it's got it in its cache, and that entry has not expired. And if it has not expired, it instantly sends back a UDP reply to me, one little packet coming back, that says this is the IP that I have in my cache for GRC.com. And my machine then says, okay, fine, and then it establishes a connection using the IP address, having had it looked up for it by my ISP's DNS. Now...

Leo: I'm really surprised, I didn't realize that the hosts file was consulted first. I see the value of doing that. But boy, usually when you think of a cache, the point of a cache is to speed queries so you don't have to make that hard drive access.

Steve: Right. Well...

Leo: The queries are fast enough, obviously.

Steve: It might be that the cache is between, from an architectural standpoint, it's sort of moot. But it could be that the cache is - that the cache holds the result of either the hosts...

Leo: Oh, it might do that. I see what you're saying.

Steve: ...the hosts query or a remote fetch.

Leo: There is, I know on UNIXes - Unices? - there is a configuration you can set to say whether you look at the hosts. But the default is the hosts.

Steve: Yeah, well, see, the reason I'm assuming that it checks the hosts file first is changing the hosts file generates an immediate change in DNS.

Leo: Right.

Steve: Whereas if it were hiding behind a cache, then - oh, and actually, Leo, it can't be, because the hosts file doesn't have any notion of expiration, and the cache is all about expiration.

Leo: Right.

Steve: So a change to the hosts file takes effect immediately, whereas - and there's no notion in the hosts file of, okay, these things have a certain amount of expiration.

Leo: I wonder why even have the cache, then, if you were going to always check - I guess because most results are not in the hosts file. Not anymore. Not anymore. They used to be.

Steve: Yes. And most users have a null hosts file. They've got an example hosts file, but nothing actually in it. So, and again, the danger of that is its obsolescence. If, for example, you put - and you could, you could put www.Microsoft.com and Microsoft.com's IP in your hosts file. That would prevent you from ever having a lookup delay when you're putting www.Microsoft.com into your browser. It would instantly be provided by your hosts file, much more quickly than any sort of network traffic out to your ISP, even if the ISP had that IP in its cache. The problem is, first of all, Microsoft has a bunch of IPs that are delivered in round-robin fashion, which is one of the features of DNS is you're able to give DNS name servers a list of IPs for a given domain, and the DNS server itself will automatically rotate those to sort of distribute the load out among a number of IP addresses. The hosts file has no such facility. But more importantly, if Microsoft ever changed their IP address, there's no way for them to notify your hosts file that it's now obsolete.

So anyway, so DNS works. And basically it's been well thought through. So imagine now that we make a request to our ISP's DNS server that is either not in its cache, or its cache has expired. That is, the entry it has when it looks, it realizes, oops, this thing has gotten stale. That prompts it to essentially go upstream, that is, the ISP will have a cache for - may not have GRC.com in its cache, but it will certainly have the .com servers themselves because there's a huge number of top-level domain servers, the .com servers. So one of those will probably be in its cache because that's going to be asked for all the time. Anytime any domain expires in the ISP's cache, it's got to go to the com servers, in the case of a .com or .edu, .gov, .mil, .whatever. So the com servers are the ones that are - the term is "authoritative." They're the authoritative - they have the authoritative records, for example, for all of the .com domains - GRC, Yahoo!, Microsoft. Well, let's see, TWiT, you're in the TV domain.

Leo: Yeah, which is Tuvalu runs that one.

Steve: But Leoville.com, so there's Leoville. So the com servers have the authoritative records for all of the .com domains. So the ISP's server is able to ask one of the .com servers, hey, what are the name servers for GRC.com? Because what the .com servers have is the name servers for GRC. So the com servers tell the ISP's name server, here's the two name servers. And they might be, for example, NS1.GRC.com and NS2.GRC.com if I'm hosting my own name servers. And because - there's sort of a chicken-and-egg problem here. We're trying to get the GRC.com IP, except that the name servers are in GRC.com so there's something called "glue." Those are additional records that can be provided. So the .com server will say, oh, and the IPs of NS1.GRC.com and NS2.GRC.com, those two name servers are as follows. And so they'll provide the IP addresses of those. So now the ISP has the IP addresses of GRC.com's name servers. It's able then to ask, for example, for the IP of www.GRC.com, which it then stores in its cache for the length of time that that record has before it expires. And if they went to GRC.com, my server, for example, says I don't know, probably like eight hours or something, or maybe 24.

One of the cool things that it's possible to do, if I knew, for example, that tomorrow I was going to be changing my IP addresses, I could deliberately bring down the so-called "TTL," the Time To Live, on GRC.com's DNS records. That would have two effects. It would mean that as GRC.com's records expired in various places on the Internet, and the records were refreshed, they would be getting new records with shorter TTLs. Well, that would increase the load on my DNS server because basically it's making all of the various servers come back to me much more quickly. But the tradeoff is, when I make a change to that information, it will get out into the Internet much more quickly.

And we've all, I'm sure, there have been many times we've talked about DNS propagating, this idea of it'll take a while for DNS to propagate. Well, that's really what that means is it's something that the DNS server itself, the authoritative server for a domain, is able to control because it's able to specify how long it wants its records to live out on the Internet. And so, for example, after making an IP change, I would probably go back to a 24-hour TTL, which then when people came back relatively quickly because I had had a short TTL, they would then be getting this 24-hour TTL. And they'd go, okay, fine, we can confidently deliver these records to anyone who asks for one full day, after which we'll come back just to make sure nothing has changed.

Leo: Of course you want to balance - it's a balance between updates and bandwidth. You don't want to overdo it, either.

Steve: Yes, exactly. So, yeah. And in fact there was - actually I remember a mistake Microsoft made where they inadvertently had a very low TTL on some of their records, and their DNS servers were being slammed by the entire Internet. They were a very popular - I'm pretty sure it was Microsoft, a very popular domain. And so...

Leo: Now, explain to me why they'd be slammed. Wouldn't that get uploaded to the main servers?

Steve: Well, yes. They were being slammed, I think the TTL was down in the order of a few seconds, though.

Leo: Oh, geez. So they were getting hit by the DNS servers asking for updates.

Steve: Exactly. They're getting hit by all of the ISPs saying, oh, look, you're telling us that your records are only good for five seconds, so we've got to ask you again.

Leo: Wow.

Steve: We don't want to ask you again. We know you don't want us to ask you again. But you're saying these records are this short-lived. And...

Leo: So the ISPs will go directly to Microsoft instead of to the big 13 domain name servers. They'll actually go to the person providing that information.

Steve: Well, yes. And that's critical to our discussion. So what they'll do is, what they need to get from the com servers, for example in the case of Microsoft.com, they need to get the name and the IP address of Microsoft's name servers because it's Microsoft name servers that have all the information about Microsoft. That is, you know, www.Microsoft, any other subdomain, secure.Microsoft.com, any subdomains. And even Microsoft.com itself, even that, that IP for Microsoft is different than the name servers, which exist on their own separate IPs.

Leo: Okay.

Steve: So the only thing that the com servers have are the name server name and IP. And when you think about it, Leo, I know you've registered domains before.

Leo: Yeah.

Steve: When you register a domain, what you give it is name servers. You give it - you say, okay, it's NS1...

Leo: Here's where my name servers live.

Steve: Here's where my name servers are. And so then whoever's maintaining the name servers, they have established the DNS records in those name servers, which ISPs' DNS servers then query. So the only thing that is in the top level domain, the .com, .edu, and so forth, are the IPs of the name servers. And then the DNS servers go to those name servers in order to get the specific details.

Leo: Interesting. Okay.

Steve: So, okay. Now back in the early days there was really no concern for security. I mean, the Internet was small enough that you had the whole Internet in one hosts file. And we've talked about how back then you had people in white lab coats who made sure they wiped their feet before they went into the computer room, and there was .gov, .edu, .mil, I mean, there were some high-level domains. There were just basically a number in the hundreds or thousands of machines in total, total IP addresses on the Internet. There just weren't that many. So they made DNS so that it worked, but there was no, I mean no concept of security.

When I was thinking about this yesterday, how I was going to drive home the fact that there was no concept of security, I thought of a perfect example. And that is SSL, that we've talked about many times, the Secure Sockets Layer, and HTTPS. That was all added on afterwards. There was no security in TTL at all. That was the Netscape guys that came up with the first SSL specification, adding that way after the fact to TCP. So it wasn't even possible to have an encrypted TCP connection. You couldn't send email that was not in the clear unless you encrypted it yourself and then had somebody decrypt at the other end. So there was just no notion of this. They were amazed it worked at all, rather than worrying about how bad guys could attack it and make it not work.

Leo: It's very telling, though, I mean, the whole Internet was designed pre-security concerns.

Steve: Yes.

Leo: And this is kind of such a fundamental part of the Internet. And they just weren't thinking. They were thinking about a lot of other things, obviously. They did a good job designing it, just not secure.

Steve: Oh, I mean, the documents that describe what I'm talking about today, it's RFC 1034 and RFC 1035, were written in November of 1987, 21 years ago.

Leo: Why do you think they just - was that just the general thing was that nobody was thinking about security? I mean, just nobody did think about it, I guess.

Steve: Well, there weren't any bad guys. I mean, the bad guys all came along later.

Leo: Why would anyone mess with their nice little system here?

Steve: Yeah, and there wasn't any commercial use of the Internet. It was all government and edu and research and just sort of, oh, look what we can do. We're able to send packets off, and they get to where we aim them. Which was amazing to these guys, I mean, and should be. It was great technology. But there just, I mean, it was fundamentally about getting it to work. And whereas now we think, oh, well, of course it works. We wish just that it were bulletproof. It's like, okay, back then just making this whole DNS thing work was amazing. So when a DNS query is sent out, any kind of a query, we've talked about how IP ports and IP addresses work. You have a port at each end which is just a number from one to 65,535, and an IP address which is a 32-bit number.

A DNS server at an ISP is pretty busy. It's sitting there, and it's receiving queries from all the ISP's customers. And it's checking its cache to see if it knows the answer already, and if it has not expired, if it knows it. And if so, it sends the answer back. If it needs to make a query to another server above it, to like a .com server in order to find out where Microsoft.com's name servers are, what it does is it uses a 16-bit value which is called the "transaction ID," or also sometimes known as the "query ID," or "QID." And the idea is that that was in the original implementation, was just an incrementing counter, that it would simply count up, and it would be a larger value each time an outbound query was made. And so the idea was that when the answers came back, the DNS server would use this query ID as, like, a serial number to match up the response with the request. And that was essentially used as its own housekeeping mechanism.

Now, a long time ago, and this is, like, way predating our current dilemma, as people began to recognize that there were bad guys on the Internet, some of this infinitely trusting nature of especially DNS came - people realized, okay, DNS is way too trusting. The reason is, it was easy to spoof DNS. If you wanted to know at any point where DNS's counter was, you could simply cause the ISP to send your own name server, your own server a query. You would see where this transaction ID counter was and then be able, essentially, to spoof your own reply. You could cause the DNS server to have an outstanding query, and then you send it the answer before the actual response gets back. Basically you beat it to the punch. And since this was just a simple incrementing counter, and you could get it to ask you a question, you could determine where the counter was and essentially send a whole bunch of replies really fast after you know that you had caused the ISP's server to emit a query and essentially spoof the result.

Now, one of the reasons this was so easily done, unfortunately, is the use of UDP because, unlike TCP, where this three-way handshake confirms the IP address at each end, remember we've talked about often how you cannot spoof a TCP address, that is, a TCP/IP, because in order to set up the connection you must have packets make a round trip in each direction. That's part of what the whole TCP handshake is doing. Whereas with UDP it's just a single packet. Off it goes to the server, and then back it comes. But there's nothing to authenticate that packet. So if your system has full raw sockets...

Leo: Whoa, the raw sockets.

Steve: Ah, yes. If your system has access to raw sockets, you're able to build your own UDP packet and put any source IP in it that you want, not the source IP of your machine, but the source IP of the server you are spoofing. And you can send that packet out, and you can send as many of them as you want to, even like with a range of these query IDs, or transaction IDs, hoping - because this DNS server is busy, it's sending a bunch of queries out. So even if you're able to read the current state of the counter, it might be making other queries before you're able to get it to make the query you want. So the counter might have moved forward a little bit. But still that creates a window that you can pretty much guess where its counter is going to be. So you flood it with fake replies, and essentially you're able to poison its cache. When we talk about a cache, a DNS cache poisoning attack, that's what this is.

So one of the things that they realized was, okay, DNS has been too trusting. We're not going to use linear counters anymore on our transaction IDs. We're going to use a pseudorandom number generator so that the transaction ID jumps all over the place. So that substantially strengthened DNS. That made it much stronger. It meant that you had to guess, well, basically it meant that you had no information about what the transaction ID was going to be, assuming that it was using a good random number generator. It just meant that it was issuing queries from using any 16-bit transaction ID following a path that you could not guess. So that made it much stronger.

On the other hand, networks got faster. Computers got faster. And remember, a 16-bit transaction ID is only 64K things. For example, we would never want to trust our security to something that was 16 bits. It's just too guessable. And we've seen from the way a birthday attack works that the chance of two things colliding - remember the birthday attack is in a room with people who all have birthdays, by definition. You may not be able to guess one particular person's birthday, or if you had a given date it might not be that anyone in the room had that. But the chances of two people in the room having the same birthday is much higher because of all the number of possible matches. It's the number of people squared. It's NxM-1. So the point is that even a random number generator that's only 16 bits long isn't random enough.

Okay. Now, two things are required in order for a reply to be matched. Not only must the transaction ID match, but the packet must return to the same port it was emitted from. The TCP stack itself, the IP protocol, if you send a UDP query, a UDP packet out, you're then listening for a reply to the same port. Well, originally DNS servers ran on port 53. That is, they always are listening for queries on port 53. That's the universal DNS port. In the same way that the web is port 80, and SMTP is 25, and POP is 110, DNS runs on port 53. That is, all DNS servers listen to queries coming in on port 53. That's the DNS service port. So, for example, when my computer is asking my ISP to do a lookup for it, it sends that UDP query to port 53. But the source port, that is, the outbound port as opposed to the destination port, that can be anything.

Well, originally DNS servers, often they would either - they would just emit their own traffic from port 53, or they would just, when they got fired up at boot time, they would allocate a port, and all queries would go out from the same port. So that meant that replies had to come back to the same port. But that meant that the only thing that the spoofer had to guess was the transaction ID. Dan Bernstein, who's a well-known, well-respected security researcher, he said a long time ago that, you know, having a fixed query port, that is, a fixed port from which queries are emitted to other servers on the Internet, is insecure. He wrote a DNS server that did what's called "query port randomization," meaning that deliberately, every time the server wanted to make a query, it would open a new port and emit the query from that port, so that the response had to come back to the matching port. Well, as we know, ports are 16 bits, one to 65535. So essentially doing query port randomization and transaction ID randomization gives you 32 bits of entropy and makes spoofing DNS vastly more difficult.

Leo: Ah, very interesting.

Steve: I mean, it is 64,000, you know, 65,536 times more difficult, and virtually impractical at that point. The problem is that most DNS servers - Dan Bernstein's DNS servers, I think it's djbdns, have always done this, have always done query port randomization. The OpenDNS servers have done it. Sort of high-security DNS servers have done that. However, a couple things caused that some problems. Depending upon the configuration, for example, of corporate or ISP firewalls, it may not be convenient for them to have queries going out on all ports because that means you've got to have replies able to come in on all ports.

So, for example, sometimes a firewall policy will restrict DNS to a given locked-down port. One of the other things that happens is sometimes NAT routers can derandomize DNS. That is, you might have a good DNS server with really good query port, or query source port, randomization. Yet when it goes to the NAT router to traverse outside, the NAT router says, eh, we're going to, you know, NAT routers often remap ports. It might make them linear. It might just say we're going to use the next port up from the one we used last time and linearize it. Well, as soon as you've got ports being linear, again, they're easy to guess. You can guess what the next port or the next range of ports is likely to be and send your spoof UDP packets there. So essentially one of the things that was done by some DNS servers was this query source port randomization, which essentially made spoofing DNS 64K times more difficult and virtually impractical. So...

Leo: A good thing. A very good thing, indeed. And it's interesting because you mentioned a couple - Randal Schwartz said that's one of the things that makes OpenBSD so secure. You mentioned OpenDNS. Do they use their own servers?

Steve: Don't know whose servers they're using. But I know that from the beginning they were...

Leo: They were secure.

Steve: Yes, they were secure because of this.

Leo: But almost nobody, I mean, Cisco, Microsoft, even BIND, the traditional Free Software Foundation DNS server that's used by almost everybody, all had this vulnerability. None of them were using randomized.

Steve: Yes. Some of them had the opportunity. Some of them, I mean, it was an option that was often not used. And it's interesting because many times in the last few weeks I've heard people lamenting that Dan Bernstein was right. We absolutely had to have query source port randomization in order to thwart this kind of attack.

Leo: Very interesting. Let's talk about the attacks, in fact, this attack that Dan Kaminsky found, because he's now revealed how it works. And now that you understand...

Steve: Oh, but actually he has not revealed how it works.

Leo: Oh, others have - well, and that's the thing. He may say, no, you were close.

Steve: Yeah, but I'm going to tell our listeners what it is that everyone believes, and what it is that was found.

Leo: Yeah, it's very interesting. And we are starting to see these attacks. And that's what's really discouraging because even though patches exist now for all the servers, many ISPs aren't applying them. All right. So let's talk about the attack.

Steve: Yes. Okay. So what we believe is that Dan - okay. Basically everything I've said so far, everybody has always known. I mean, that's what DNS spoofing is. That's DNS cache poisoning attacks.

Leo: And but it was more theoretical because it was complicated and difficult to do; right?

Steve: Well, the transaction ID was normally randomized, but the source port for most DNS servers was not. So but it was believed that even if, I mean, the worse you could do would be to affect one record in a DNS server. That is, it would only be when www.Google.com, it would only be when that record expired that the server that had cached it would be induced to go and update its knowledge of the www machine IP at Google.com. So there's a tiny window of opportunity. There was nothing you could do to, like, force an ISP's DNS server to accept a new www.Google.com IP. There would be - when it would finally expire, then you had an opportunity. But, I mean, it would immediately be asking. And you had transaction number randomization. So there was just - it wasn't practical. It was sort of a theoretical attack. It wasn't practical. What we believe Dan realized, what now exists in exploit code being circulated around the Internet, HD Moore, who maintains the well-regarded in hacker circles, the Metasploit Framework, instantly implemented this late last week in his framework. And he called it the BailiWicked Domain Attack.

Leo: I like that.

Steve: "Bailiwick" is the term used to refer to the sort of the neighborhood, the domain that you're in where once upon a time it was possible, in early DNS servers, to fool them by sending information back that they did not ask for. That is, you could actually, the way a DNS query and reply are structured, you have a query and then an answer to the query and then additional information and authority information, four sections. And you're able to put whatever you want to in the additional information. Normally, it's as I was saying, it's this glue to sort of help the server. For example, if the server you're querying has some additional information for you, like the IP addresses of the name servers or other information that you might otherwise have to go get again, it'll give it to you to sort of, just for the sake of efficiency, to improve the way DNS operates. And there were some servers where you were able to give it - and this is early DNS servers - you were able to give it information about an entirely different domain. So that in a response to a query about Yahoo.com, you could actually put in information about Google. And this has not been seen for a long time, so don't worry about this. This got fixed quickly. But, I mean, this is how trusting the original DNS was. It just believed anything anybody sent it.

Leo: Whatever.

Steve: And so those were called "out of bailiwick," that is to say, the additional information had nothing whatsoever to do about the query. They weren't in the same bailiwick as the query. So now DNS servers were made smarter, and they were taught not to, well, to completely disregard any additional information that is out of bailiwick, that is not relevant to the query that they're making. So what we believe Dan realized is that there was a way to force, now, as opposed to waiting for the cache to expire and then having some futile race in order to get a reply in before the real reply was able to get in, we believe he realized there was a way to force a fundamental change in the DNS cache.

And this is how it works. You first - let's use - I'll use Example.com. So you look up the two name servers for Example.com. And that'll be NS1.Example.com and NS2.Example.com or whatever they are. You look up the two name servers for Example.com. Those are the servers that an ISP's DNS server will query when it needs information about Example.com. So you're able to look them up just the same way the ISP is. So you know the IP addresses that the ISP's DNS server will query. And so you're going to attack this given ISP's DNS server. And this will be a lame ISP that has not updated their DNS servers to add random query port queries.

Leo: A surprisingly large number, even now.

Steve: Yes. Even now we're seeing a huge number that have just not bothered, even though it's been - there's been a lot of concern raised about this. So what you do is you send a query for a fake machine, Aaaaaaa.Example.com. And, okay, that doesn't exist. So you know it's not in the cache. That is, you're forcing a cache miss, which forces the ISP's DNS server to ask the Example.com server for the IP of Aaaaaaa.Example.com. So what happens is you know that upon making that query to the ISP server, the ISP server is going to send an outbound query to get the machine, the IP address of that wacky Aaaaaaa machine. So you know that there's a query outbound. If they're not using source port randomization, you know the port from which the query was emitted. And even though they're using random transaction IDs, 64K is not enough. So you flood it with maybe 100 UDP packets. You spoof your source IP as the IP that you know it's asking, the IP for Example.com's name server, which you were able to look up. And you know the port that it's bound - you know the port that you send it to. So you flood it with replies. Even if that misses, it doesn't matter because now you ask it for Aaaaaab.Example.com. And you flood it with more spoof replies. And then Aaaaaac.Example.com. Essentially what we've done is we're able, by making up machine names at Example.com and querying an ISP's DNS server, we're able to force it to ask Example.com for the IP. Now, our spoof replies include new name servers for Example.com. And because they are name servers for Example.com, they are in bailiwick, and they will replace the name server records at the ISP.

Leo: Wow.

Steve: And it works. It takes about 30 seconds, maybe as much as a couple of minutes. It is being exploited right now, and it works. What this means is you're not just changing one record. You are essentially redirecting all the name services for an entire domain - Google.com, Microsoft.com, eBay.com, Yahoo!, I mean, whatever the attacker wants. They're able to flood an ISP's server with spoofed replies. Even though the transaction ID is jumping all over the place, the nature of the birthday attack means that you're going to get a collision. When you get a collision, it's going to look like a valid reply from the real server. And that real server, the so-called "glue," this additional information will be accepted because it's in bailiwick. In the process what you've done is you have told the ISP server that these are the name servers for an entire domain, which is where it will go from now on.

And you can give that record a super-long TTL, a super-long Time To Live. Time To Live is 32 bits. That's a long time. Essentially, it will never expire. That record will never expire, and it will sit there being wrong. At that point anybody who is a customer of that

ISP who is using that ISP's DNS and puts www.Google.com, for example, into their browser, their machine will receive a fake IP for Google.com. It'll come up, and it'll look just like Google. You look in your URL, you can make sure, oh, yeah, www.Google.com, that's it, there' s no phishing going on. Wrong, because the ISP's DNS has been poisoned with this attack.

Leo: And none of the anti-phishing measures that the browsers implement will work because it looks like it's the normal place, you're going to the right place.

Steve: Yes. Now, what does work is...

Leo: Certificates work.

Steve: Certificates, yes, SSL. There is no way that some random malicious server in Russia that is spoofing Google or spoofing eBay or spoofing PayPal, and these are going to be the targets, there's no way that they're going to be able to have a certificate for PayPal.com. However, not all users are cognizant of security. That is, it's PayPal, when you go to www.PayPal.com, PayPal switches you to a secure connection for login. We just take it for granted. And it's only the most recent browsers that even color the bar and show you that you've got a secure connection. No, it's a little padlock down in the bottom. People don't look. And they just assume PayPal is going to, when they log in, it's going to be secure.

Leo: It can never be a secure connection because the certificate doesn't match the URL. You don't own - you can't get PayPal's certificate.

Steve: Well, and my point is that the spoofed site will not switch you to SSL.

Leo: It can't.

Steve: But, see, it doesn't want to. It doesn't need to. Because you put in www.PayPal.com, which by default is http. So you establish a connection not using SSL. It gives you the login page. You think you're at PayPal and that it's going to be a secure login. But it's not because you're actually at the wrong IP, which every customer of this ISP will be going to as long as that cached DNS record is wrong.

Leo: And they couldn't really ever make it an SSL connection, a certificate, because a certificate would have to - would it have to say "PayPal.com" to match?

Steve: Yes. But, see, they don't have to. I mean...

Leo: Well, I understand they don't have to. But what this means though is there's a hole because I can check to see if I have an SSL connection. And if I don't, then I know it's not the right page.

Steve: Oh, absolutely, Leo. You and our listeners are probably not going to get caught out by this.

Leo: Well.

Steve: And so you're right. You could...

Leo: Have to remember to check.

Steve: ...put in https://www.paypal.com. If you manually put in a secure connection, PayPal, the real PayPal, will accept it. A fake one, well, they're probably not even going to have a server running on port 443 for https because they know they're not able to spoof PayPal's certificate. But they could do something even trickier. They could, at www.PayPal.com, they could have a redirect. Remember we talked about the redirect dance a couple weeks ago that Phorm was doing. They could have a redirect to PayLal.com, some subtle misspelling of the name, and they'd get a certificate for that. So they're able to accept a connection on PayPal, redirect your browser to https://paylal.com...

Leo: But it would say PayLal in my browser. It wouldn't say PayPal in my browser.

Steve: Right, or it could be PoPal or something, you know, like "O" versus "A," where at a glance it looks the same. But again, if there were some need for them to establish an SSL connection, they have, I mean, when you go to the site nonsecured, they can do whatever they want to with your browser, which believes it is at the right location.

Leo: Right. That's fascinating. So if you are sharp-eyed, you can spot this. There is a way to always spot it. Well, I guess if you're going to a site that has SSL.

Steve: Yeah, I mean, a blogging site or all, I mean...

Leo: You wouldn't spot it.

Steve: Unfortunately SSL connections are still the rarity. They're not used except normally only briefly during login, even, for example, when you log into Gmail, as we've talked about. If you initiate a connection to Google Mail that is secure, it'll keep it secure. Otherwise it briefly moves you into secure and then back into nonsecure.

Leo: Very interesting.

Steve: So it is, I mean, this is a - and this is serious because, first of all, HD Moore has implemented this in the Metasploit Framework. These attacks are underway. They take on the order of a couple minutes to get the collision of the transaction ID on a server that is not randomizing its queries. And unfortunately a huge number of servers now are not. Now, Dan created a page that we talked about last week and the week before, I think, his DoxPara site. He's updated it a few times. I've seen and others have reported inconsistent results from it. Sometimes it says all the queries are coming from a single port. Sometimes it says they're coming from random ports. I've even seen it say that your DNS server is doing something better to thwart the attack. You don't have anything to worry about. But he's being cagey about it.

So there is another site that I wanted to tell our listeners about. It's got a long URL which I am putting in our show notes. I'll just say it for the sake of saying it once. But I've created a SnipURL, a short version of it. The full URL is entropy.dns-oarc.net/test.

Leo: But don't write that down.

Steve: You don't need to write that down. The easy-to-remember one is snipurl.com/dnstest, which is just a little redirection URL that I created for our listeners. What that does is take you to a very nice test that actually charts the source port randomization and the transaction ID. It issues up to 25 queries and watches the DNS server that your browser is using, which is typically your ISP's DNS server, it watches it issue queries and is able to show you a chart and lists all the ports that were used and all the transaction IDs. It then does a standard deviation of those.

Leo: It's amazing, really. I mean, if you looked at the old DoxPara, it's incredible.

Steve: Yeah. This is a really nice tool, and it will allow people to determine if their ISPs have figured out that this is really important to fix. And we should mention again that the OpenDNS system, which has always been available and for use as an alternative DNS, you've got to manually configure it in your system, but they're using source port randomization, and they're not going to be spoofed. And it might, if you're a person who's concerned about this, you might want to take matters into your own hands until your ISP gets around to fixing this.

Leo: Yeah, OpenDNS is great for so many other reasons, too. I think that some people have worried a little bit about the notion that they put ads up if you get a, you know, you enter the wrong URL. Doesn't bother me. They have to monetize somehow. And it's so useful in so many ways. And it's secure. and it was secure from day one.

Steve: Now, there have been some questions about whether personal routers have a problem with this. And personal routers for the most part are not doing any caching. The little inexpensive Netgear, Linksys-style routers, all they're doing is passing DNS queries on. However, the Linux-based routers, like the OpenWrt stuff, those are known to have this problem. So people who are using the ww-wrt and OpenWrt, you're going to want to do some Googling and find out what's going on because those routers may be caching. But again, it would be - it would have to be a highly targeted attack. That is, somebody would have to see your router issuing a DNS query and have some reason to poison you.

In the process, they're only going to poison your one network. Of course, you're concerned about that. Except that it makes far more sense for bad guys to go after major ISPs.

And essentially what this means is they're able to insert new name servers into any domain they want and poison for any length of time that domain. And the only way an ISP would probably know is if customers began complaining that when they go to PayPal something seems wrong. And then the ISP goes, uh-oh, and flushes that cache record and gets rid of the problem. But who knows how many people would be affected in the meantime? I mean, this is why all of the DNS solution providers updated their DNS servers. They may be - we don't know what the full fix is. That hasn't been revealed. We don't really know for sure what Dan is going to say on August 6th. Maybe they are tightening up their in-bailiwick management. So, for example, they're looking to see whether, like, another reply came back from the real server. They could be doing rate limiting on incoming spoofs. I mean, there are a number of things DNS could do to harden itself even beyond source port randomization. But certainly adding another 16 bits of randomness by virtue of emitting queries from an unknown port, that is just much, well, it's virtually impossible. Well, again, "impossible" as we now know is a relative term.

Leo: Never say "impossible," yeah.

Steve: Never say "impossible." But it's so unlikely that both the transaction ID that's 16 bits and the port number which is 16 bits minus one, so unlikely that both of those would match an incoming reply as to not be worth anyone's while trying to guess it. So certainly the best defense is to have an ISP whose DNS servers are emitting queries on random ports. But I think that this point our listeners probably know and understand, hopefully, this entire problem.

Leo: Wow. You know, I had read the explanations that others had come up with about how this works. But you've got to have the fundamentals of how DNS works before you can understand this. And I didn't really quite understand this whole port thing. And now that you've explained it, I do. And the exploit makes sense. And I think the thing to point out is that we're seeing this exploit in the wild.

Steve: Yes, yes. Interestingly, HD Moore commented somewhere that this was available on Linux, and they were porting it to the Mac OS X, that is, the exploit. But it would not be ported over to Windows because Windows cannot spoof UDP because...

Leo: Because...

Steve: ...thankfully it does not have full raw sockets.

Leo: And this was a fight, just because I know Steve won't toot his own horn here, this was a fight Steve brought to Microsoft when XP first came out and it did, in fact, have this raw sockets capability. Microsoft put it in saying, well, it's in UNIX, we ought to have it in Windows. They underestimated the security issue. Steve really raised a ruckus. A lot of people, I have to say, criticized Steve for that, saying oh, it's a tempest in a teapot. Well, he's been proven right time and time again. And eventually Microsoft finally got the message and did take out that full raw socket capability from XP. And thank goodness. Otherwise these exploits would be much more widespread.

Steve: Well, what we would have is we would have Windows-based bot fleets that were able to do this. And that would really up the ante. It's also worth mentioning that Dan has come under a lot of heat. People are accusing him of promoting himself and making a bigger deal of this than it is. I mean, I think there's no way not to feel that Dan has really done the Internet community a great service. He realized this was a problem, presuming that this is what the problem is. Again, we're all having to really finally wait until his presentation at Black Hat to know. But he brought this to the attention of the proper DNS authorities, got them to really explain the problem, got them to understand that this wasn't just a theoretical problem. Everybody fixed it, or that is to say, changed the software so that it could then be deployed.

And then, as we announced two weeks ago, this thing all went public. Not any knowledge of what it was, but just the fact that people had to upgrade their DNS. And maybe they're strengthening DNS even beyond good source port randomization. We really won't know until Dan tells us the whole background of this. But it is certainly the case that old DNS servers are still on the 'Net. I mean, tens of thousands of them are. And tools now exist that within a few minutes can change the DNS records of any domain they choose, to redirect everyone who uses that, who depends and relies on that DNS server, to any other IP on the Internet. And that's not good.

Leo: Well, and he handled it so responsibly. I mean, I don't think you could ask for a more responsible way to handle this.

Steve: No.

Leo: So no reason to criticize Dan in the least. In fact, he did, it seems to me, exactly the right - and there's no way he could have known that ISPs would be so slow to adopt these patches. And what are you going to do? You've got to make - you've got to do it. Anyway, soon as the patches go out, people know something's up.

Steve: Well, and now what'll happen is we will - ISPs, for whatever reason they're being reluctant, they don't want to bring their DNS servers down, the guy with the password's quit and they don't know how to bring them down, I mean, who knows what. But we're now going to see some horror stories, isolated horror stories of major exploits of this against unpatched DNS servers. I will say again what I said last week. It is very easy to determine if a DNS server is exploitable. So that means that those servers that have not yet been updated are going to - they're, like, drawing fire to themselves.

Leo: Is there any way to tell? I mean, yeah, you could run one of these programs, I guess, like DoxPara or DNS Test, and you could tell.

Steve: All you have to do, yes, all you have to do is cause the server to emit some queries...

Leo: Oh, you can see it immediately.

Steve: Yeah. You're able to see - you would set up a domain, and you would ask it for fake machines in your domain. That would force it to ask your domain servers for those fake machine names. You look at those, you instantly know whether that DNS server is exploitable or not.

Leo: Wow. Very interesting stuff. Thank you, Steve Gibson. You can find this podcast, plus a transcript, plus 16KB versions if you'd like to share it with your bandwidth-impaired friends, at Steve's site, GRC.com. That's also where you'll find SpinRite, the world's best, must-have disk maintenance and recovery utility. It gave me a great feeling to know I had SpinRite and I could test this drive when it failed. GRC.com. Also a lot of great, useful utilities like ShieldsUP!. Steve's really been doing this for a long time and is a great boon to the Internet community. And we're really glad he's there. GRC.com. Steve, we will do a Q&A next week.

Steve: I look forward to it, Leo. I'll talk to you then.

Leo: All right. Thank you, Steve Gibson.