ICANN86 Seville | PF – DNSSEC & Security Workshop
Monday, June 08, 2026 – 14:45 to 16:00 CEST

KATHY SCHNITT

Hello and welcome to the DNSSEC and Security Workshop. My name is Kathy and I'm joined with my colleague, Dan, and we are the participation managers for this session. Please be advised that this session is being recorded and is governed by the ICANN Community Participant Code of Conduct, the ICANN Expected Standards of Behavior, and the ICANN Community Anti-Harassment Policy. Interpretation for this session will include English, French, and Spanish. To help us keep the discussion moving smoothly, here is how you can get involved today.

To keep the queue fair for everyone, all participants, including those here in the room, should use the raise hand feature in Zoom. When speaking, state your name and affiliation for the record, the language you will speak if speaking a language other than English, and maintain a moderate pace.

You may also submit any questions for the speakers today through the Zoom Q&A pod. We will read those aloud as time permits and when directed by the moderator. While the chat is available for attendee interaction, please note that the comments or questions posted in the chat will not be read aloud. Official questions must be raised through the speaking queue or submitted through the

Q&A pod. And with that, I'm happy to hand the floor over to Dan York.

DAN YORK

Greetings, welcome. Thank you, everybody, for being here on this day here at ICANN86. This is the DNSSEC and Security Workshop. We have been doing these for, I don't know, 15 years now, in different forums at this event. They continue to evolve and take place. I want to first thank our speakers who are going to be talking to us today and also just extend the invitation that the program committee is always looking for more speakers for these kind of sessions.

This is a compressed format because of the way this policy forum is set up. We have only this one session today. We anticipate having a longer session at the next one at ICANN87. And if you're interested, if something here causes you to have some interest in presenting something you're doing, please look for the call for proposals or we will be putting that out soon for the next ICANN87.

So today, in the brief period of time that we have, we will be having a presentation first from Peter over here on the far side. He'll talk, as you can see, on the agenda here. Wes Hardaker is then going to be coming in and talking about, he tells us it's the more interesting of these presentations, injecting a man in the middle in front of a validating resolver. Wes is currently, unfortunately, sick in his hotel

room, but he has improved enough to be able to come and present to us virtually.

And then we have Graeme Bunton, who's sitting next to me here, to talk, to give us a sense of what NetBeacon is all about. And then finally, the last slot may be Wes, or it may be Warren, who is here, who will be on something completely different. So we will see. Thank you for journeying with us on that. And so with that, I will turn it over to Peter.

PETER THOMASSEN: Yes, hello. My name is Peter Thomassen from deSEC. I will be giving a presentation on misconfigurations of DNSSEC and of HTTPS. So let me jump right into it. This graph, courtesy of Wes Hardaker and Victor Duchovny, shows the percentage of secure delegations that are misconfigured and don't validate. So this is from dnssectools.org. And you can see that it's roughly between 1% and 2%, with a bunch of spikes, I guess, when various things happen. Anyway, so it varies. And yeah, so that's roughly the level. I think it's an interesting question to consider what the audit rate of other security technologies is.

So DNSSEC is error-prone, 1% to 2%. I just showed that. There's many reasons for that. It could be a key missing that is referenced in the DS record set, could be wrong signatures because the wrong key was used, could be expired signatures, could be missing. And then you get things like on the bottom right, which is a terminal

screenshot that shows an error condition where the DNS key doesn't match the DS record set.

Okay. So for HTTPS, we don't know how error-prone that is until we measure it. There's also a number of possible reasons for that. So it could be the wrong domain name in a certificate. It could be that the certificate was issued by a certificate authority that is an intermediate on the way to another trusted one. So then you have to include that intermediate certificate in the handshake. If you don't do that, an error occurs.

Then the main CA could be unknown. It could be expired and so on and so forth. What we do know is, or at least feel like, that that was more frequent before automation was introduced with the ACME protocol and let's encrypt. But yeah. So how common is it really? So I decided to do a measurement.

The inputs to the measurement are the crux top million lists, which is a list of domain names ranked by essentially page visits in Google Chrome. And so by web origin. So that means HTTPS or HTTP with the domain name part, but irrelevant what the path is. And yeah. So this better reflects what people actually visit as opposed to things that rank things by DNS queries, for example.

And so there is the thing that I tried was essentially trying to connect through HTTPS. Now browsers not only use the operating system just store. Sometimes they also have lists of those intermediate certificates, which sometimes are missing from the handshake. And so they sometimes bring them hard-coded more

or less to work around these misconfigurations. So I decided to also do that to simulate real browsers.

The measurement was as follows. For any domain on the crux list, I attempted a TLS handshake on port 443, and then I recorded whether that worked. So that would be okay. Or whether the certificate was expired or the certificate authority was unknown, or the domain was a mismatch, or there was some other reason for TLS handshake failure, like unsupported algorithm or other things. And another error condition is connection failure when the IP address couldn't be reached or the connection was reset or other reasons. It's mainly to distinguish from what's the TLS issue and what's some other sort of failure.

And I did that both for the operating system trust store only, which is what you would experience when you, for example, use curl or some other, I don't know, when you write a Python script that connects to an HTTPS server, it would only use the operating system trust store usually. And then I repeated the experiment with the intermediates mixed in that I mentioned on the previous slide to more accurately mimic browser behavior. That was then conducted on April 29 this year. And I did aggregate the results by effective TLD, which is essentially TLDs and second level stuff like co.uk.

So here's numbers. The pie charts at the top are not very interesting. You can see that most things succeed and then there are some issues. The graph at the bottom is more interesting. So

you can see that expired certificates at the bottom left are very uncommon today, 0.01% or 0.02%. The solid area is without the intermediate certificates mixed in and the shaded area is with the intermediates mixed in. In the case of the expired certificates, I mean, there is some statistical fluctuation always. So it seems to be the case there.

The name mismatch and some other TLS reason like unsupported algorithm or connection failure are also not very important failure modes, although they are some fraction of a percent. And of course, connection failures are more, but they are not TLS issues. You can see the numbers are roughly independent of whether you mix in intermediates, which is expected for failure modes like name mismatch.

Then if you only use the operating system trust store, about 1.8% of connections fail because the CA isn't known. And that goes down to 1.3, so roughly by a quarter if you include the intermediate certificates that browsers sometimes use to work around misconfigurations. So that tells us that roughly the failure rate is one to 2%, as in the DNSSEC case.

Here's another graph that shows the failure rates by effective TLD and they are ordered by the number of delegations essentially. So the biggest one is .com. You can see the failure rate is roughly one to 2%. There's ones that are particularly difficult to connect to like .ru domains or .in domains in India. There are TLDs which have particularly low unknown certificate authority numbers, like the

Czech Republic, for example, here has .4%, I think. And yeah, so this is an interesting compilation, I would say.

And the absolute numbers are given in this graph. It's essentially the same information as at the top. Note that the y-axis is logarithmic, so each axis tick is a factor of 10, which of course is expected as the domains on the left are much larger.

All right, then this is a graph that shows how the failure rate changes when you mix in the intermediate certificates. So the failure rates by effective TLD are the red dots. So for example, the range from around 1.8% to 8% between some Korean domain names to .online, which has high failure rates. And when you mix in the intermediate certificates into the just store, the failure rates of course go down.

And interestingly, it depends a lot on the TLD by how much it goes down. The cases where it doesn't really go down, like where the dots are the same, .site, .online, .shop, .xyz, .vip, that indicates a lot of self-signed certificates essentially, where domains probably are parked and people don't care about the misconfiguration. That would probably also be the case if they did use DNSSEC.

And then for more production grade domain names like gov in or pt, mx, the failure modes go down. So the rates go down a lot if you mix in the intermediate certificates. Interestingly, it depends a lot on the TLD how much it goes down. For example, .gov, .in seems

to have a common misconfiguration about missing intermediates, whereas .info doesn't.

All right, so the answer is HTTPS is error prone on the 1% to 2% level. So this is the same slide as before. I've just filled in the question marks. And it's roughly the same as for DNSSEC. I think this is an interesting data point. I want to make very clear that I'm not making any claims here about impact. Obviously, a DNSSEC failure of a TLD has much larger impact than an HTTPS failure because a DNSSEC misconfiguration in a TLD is roughly like a certificate authority failing so that all the domains that are under its cryptographic control experience an outage when the CA does or when the signing authority, which is the DNSSEC TLD does.

So this is mainly meant to add data points to the discussion because DNSSEC failure numbers are often circulated and HTTPS failure numbers not so often, so I figured it's useful to have these around. I'd be interested in some feedback whether this is useful. And if it is useful, whether, for example, it would be a good idea to make a website that has a timeline that shows how this develops, just like the timeline in the first slide showed how the DNSSEC error rate develops.

It'll be also interesting to see whether the DNSSEC and HTTP error rates change over deployment of automation. So if anyone wants to place a bet, I'd be interested in that. And yeah, so I don't know if you have any ideas on the measurement method or data set, how

that could be improved. I'd also appreciate input on that. Thank you. If there's questions, I'd be happy to take them.

DAN YORK Thank you, Peter. Are there any questions for Peter? Warren?

WARREN KUMARI I mean, not really a question, more of a comment. Yes, I think there should be longitudinal studies like this. I know at one point ISOC was doing one, but I guess they abandoned that or something. I think ISOC was. You all were? You had part of the certificate? You had a certificate?

DAN YORK All right. Any other questions? How many people would like to see Peter build a website with this information on it?

PETER THOMASSEN I didn't say I would do it.

DAN YORK Oh, how many people would like to see Warren build a website on this? Look at all those hands. Okay.

WARREN KUMARI They track TLS metrics through the Internet Society Post thing.

DAN YORK

Well, yes, we do that. Okay. That was a different thing. You can. Okay. Any other questions for Peter? All right. Well, thank you, Peter. All right. Our next speaker will be Wes, if he's on here. There he is.

WES HARDAKER

I should be. If my hair's sticking up or something, sorry, don't laugh at me. Warren's going to laugh anyway. All right. Let me see if I can share my screen appropriately. Click. Please work. Please work. Oh, good. Okay. You guys can all see that, correct?

DAN YORK

We do, Wes. It looks good here.

WES HARDAKER

Okay. So, this talk is going to be about injecting a man in the middle, or excuse me, a machine in the middle, except I'll be the man, so it'll work, in front of a validating resolver. I'm sure nobody has any clue what that just said, so let's dive into what we're actually going to talk about today.

So, imagine you have a network. Imagine your network has a bunch of authoritative servers scattered around the internet. Those are the green bubbles. There's stuff for ICANN, and there's stuff for the root servers. And, of course, your resolver in your local network, the blue circle in the middle right, needs to talk to them

all, right? And you're a client that needs to know how to interact with them. Well, imagine that I'm in your network, too. And imagine that I'm not very nice. Imagine that I put myself as a machine in the middle, sort of near the root server, near icann.org, right? A machine in the middle allows you to sniff the traffic along those fictitious wires, in this case.

So, it shouldn't be a huge shock that because I'm sitting in the middle of that particular path, I can view all of your resolver's DNS requests to the DNS root server or to icann.org in this example diagram. So, that's not surprising. DNSSEC prevents me from modifying it, mostly. So we're all good, right? This is what everybody knows.

Surprisingly, though, under the right conditions, I can also watch all the traffic to these other zones, to example, example.com, and org. And that should be a little bit more of a surprise because I am not in the middle between the resolver and the zones. And even the ones on the upper right, which is a completely different direction. That's what I'm going to talk about today is why this happens, when it happens, and under the very specific conditions when it does. It doesn't happen to everything.

The issue is that parent-centric resolvers, of which there are a number, and we'll go over what the difference is between a child-centric resolver and a parent-centric resolver. But parent-centric resolvers trust unvalidated data when they're making their queries to the network. I will give a demonstration of this, a very quick one.

And then, of course, I'll talk about well, given this problem, what's the worst that could go wrong, and what can you do to mitigate it?

So first off, let's go a little bit of background with resolvers communicate with everything. So this is sort of the logical DNS tree. And one of the hard things about this presentation is I'm going to be talking both about networking paths and about sort of the DNS hierarchy.

So this is the DNS hierarchy that everybody is fairly well familiar with at this point. But I need to talk about all the networking paths between a resolver and each of these points out in that hierarchy. So each one of those green, the top three rows of things are representing authoritative DNS servers, and then the resolver has to talk to them all. And they don't go through the same lines, right? One's delegation, one is network paths.

So let's talk a little bit about what actually happens, right? When you make a request from your client, your local computer or whatever, and you request www.ietf.org, and you want the address record, which is the A, it sends it to the resolver. The first thing the resolver has to do, assuming it doesn't have anything in its cache, is to ask the root, hey, where is org?

This guy wants something inside of ietf.org, and I don't even know where to start. Where's .org? And the root responds and says org is at these name servers, which are listed there as org1 and org2. Then it has to go and say, hey, now that I know where org is, I'm

going to reach out to org on the left-hand side and ask where is ietf.org within this hierarchy.

Now, the vertical lines in this are sort of representational of, given that previous diagram that I showed you, what things in the middle get to see some of this traffic. And then finally, in the end, it says, hey, what's the address for www.ietf.org, and you get the answer back, which is then eventually given back to the client. So, these network flow diagrams are kind of tricky to read if you're not familiar with them. Hopefully, they'll make at least a little bit of sense.

For now, though, I want to dive into just one of them, because this is the sort of basic introduction, and of course, a lot more goes on than just looking up the name servers for .org, especially when you turn on DNSSEC. So, this is sort of the same box of just that one .org box before. So, there's actually a bunch more stuff that goes on here.

So, when you are doing DNSSEC, you end up searching not only for the name servers, but also for the cryptographic records, the DS records. And those are, the DS record is signed, the name server records and the IP addresses and the glue records are unsigned. And that's going to come to be important here in a minute. Then a resolver can go check those records by getting the key from .org and then getting the name servers for what .org thinks is the name servers. And then checking those, because that's actually where the signatures exist.

So, what actually happens when a difference arises? What happens when the root gives back two addresses for .org and .org says, no, I'm actually, I got rid of org2, I'm now at org1 and org3, because I changed something in my network. Well, parent-centric resolvers, because there are no signatures on the original records, and because they never query the children, end up using the unsafe, unsigned records up above.

And then child-centric resolvers will do steps five through eight as well to verify that the name servers that they actually got back are correct. And even this is somewhat of a very high-level overview. And I know there's people in the audience who are going, well, there's a lot more that goes on here. You didn't show our SIGs. Sometimes additional queries are made when you're trying to debug a problem inside of a resolver. I'm keeping this as simple as I can for this demonstration today.

So why would you do one versus the other? Well, there's actually good reasons for picking one or either one, right? Child-centric resolvers really only have one cache that stores all the data, whereas parent-centric resolvers have two caches. They actually cache the delegation information separately than the child data. And I can hear groans from people in the audience that are saying, it's more complex than that. I know it is.

The child-centric resolvers double-check the child's own information. That was those last four steps that I put on the previous slide, whereas the parent-centric resolvers get the first

answer they get and say, oh, this is good. We'll use that. Child-centric resolvers, therefore, end up validating basically all of the data that they have to use one way or another, whereas in parent-centric resolvers, the only data that's actually DNSSEC validated is what they give back to the client in the very end of it all.

So this doesn't get in the way of giving the right answer to the client at the end. It only gets in the way of where they're getting the information from. We'll get back to that in a number of slides later, too. So the downside of child-centric resolvers is that they're slower. They have to do more queries. That simply makes them slower.

And then parent-centric resolvers are, therefore, faster because they do less queries. And there's been recent studies that ISE did recently, and I think others have, too, showing you actually get more correct end answers without errors. And I can't summarize their work very well, but let's just say that it's a little bit more resilient. You may actually be able to return an answer to the child that you couldn't if you were a child-centric resolver.

All right. Hopefully that all makes sense. So who cares, right? What's the worst thing that could happen? Well, I want to introduce you to my example of how we can mess with a parent-centric resolver. If you haven't seen the movie The Parent Trap, there's both a movie that was made in, like, 1970 or something quite old, and this is a screenshot of the more recent Parent Trap movie. Anyway.

So let's go back to this diagram again and look at what happens. If I request www.ietf.org, it goes to the resolver. The resolver has to go pick something up from .org. What if me, the pink machine in the middle box, stops it? Says, nope, I'm not going to pass it on. I'm not going to give it back to the resolver. Instead, I'm going to give the resolver different information. I'm going to give it different name server names, potentially, or different addresses.

If I do that, and I give it the address of the machine in the middle itself, for every other name server out there that it ever asks for, I'm always giving it mine. It's always going to come back to me. Instead of going to .org on the left-hand side or ietf.org on the left-hand side, it's only going to check with me. I can keep lying to it forever, all the way down the hierarchy.

So in the end, when the resolver finally says, okay, I've gotten through everything. I keep having your address, but I'm going to keep leaving it. Can you at least tell me where www.ietf.org is? The machine in the middle actually has to return the right answer, because this is where DNSSEC will fail if it doesn't have the right final answer to return to the client. So the machine in the middle actually has to come up with the right answer and query itself from the real servers to get it. Eventually, that is returned back to the client.

All of this is fairly complex. You can imagine saying this doesn't actually work in real practice. Well, it does, and I actually created a demo of that. If Kathy or somebody in the room, if there's any

hands going up, if you're lost or confused at some point, let me know.

I went about testing this. Can I actually demonstrate this in a real world? I created this test network that basically did everything that we were just talking about. The client and the resolver are there. The client sends questions to the resolver. The resolver has, the way I set up this test network using Docker containers, for those that care, all of the routes to the root server go through the man in the middle. So I actually injected IP routes in the resolver to say, nope, if you're going to talk to the right route, you're going to go that direction. Anything else, you can go talk to com, net, org,icann.org, whatever, through the default route. Internally, basically what I do is I firewall off all traffic to the root from the man in the middle, and I let it respond with its configured information for what it's supposed to do. We won't go into the details too much because it gets even more complex.

What does the demonstration now look like? Well, the demonstration is going to first need to know, hey, where is .org, if we query it for ietf.org, and it will go out and it'll ask that the machine in the middle will be in the middle between it and the root. It'll ask the root, hey, where is org?

The root will respond with, org is at these name servers and these glue records, these address records, and then that will return them to the resolver. That's the perfect world. In my not-so-nice-person world, I can return the wrong glue. I can return these 10-dot

addresses, which are not routable on the real internet. If I keep returning my address, the resolver is going to keep coming back to me rather than sending things out of its default route.

This continues for a while. First, we have to ask where .org is, then we have to ask where ietf.org is, and then eventually where www.ietf.org is. I'll repeat what I said earlier, which is that I still have to return the right answer to the client, or to the resolver, because it actually will check the signatures on the keys, the DS records, and the final answer. Note that I didn't say name servers and I didn't say glue. This is still only true for a parent-centric resolver. Not all of them are. We'll talk more about that later. Eventually, that answer actually has to go all the way back to the resolver, and the client will actually never really know.

All right. Let's do a demonstration, if I can pull this off, because live demos when you're sick always go twice as better as live demos when you're not sick. I have these Docker containers that do a bunch of stuff. There's a bunch of them in there. You can see that the important ones that we're going to talk about are DNS machine in the middle, and then there's also resolvers and a bunch of stuff. It basically implements that diagram I just showed you.

So, this is actually going to start the machine in the middle thing that gets in the way of everything. It started up a parent-centric resolver, and over here you'll see that there's a thing called DNS watcher that starts up, and that is basically the machine in the

middle address and what it should be using to return stuff to and things like that.

Then, I have a script that just sends the dig request to the client. So, it basically acts like it's the client. I could log into the container itself directly, but I'm just writing a script to do it more easily. And if I look at www.ietf.org, now before I do that, I want you to note that the only thing that this container should be seeing, the machine in the middle, is traffic to the root. Everything else goes out that normal path where it should, unless I'm doing something mean, which I am.

So, if I dig for www.ietf.org, you can see I got an answer. You can see that answer is even validated. For those that don't know, the 80 bit in that flag means it was validated. DNSSEC has proved it all the way from the root down. However, what you shouldn't see, but yet we do, is that all of these requests that went through the machine in the middle for segments of the internet zones that they shouldn't be able to see, right? Www.ietf.org and org should go out a different path. The machine in the middle shouldn't see it.

And we can do things like, well, if we look at the real address for rootservers.net, and we look at that, that's the real root server address. And if I actually connect to, this is the most complex bit, and then we're going to go back to slides here. What did I call it? Parent. If I connect to the parent-centric resolver, and I get a bash shell on it, I can do a dig at the real root IP address, and I can ask it, hey, what are the nameserver records, right? This is exactly what

the resolver would do, the parent-centric resolver would do. Hey, what are the nameserver records for dot?

And you'll note that all of the answers that came back are all 10 dot addresses. So I have completely messed with the root server addresses. And it didn't check. And it even says that they're secure, too, which is even more interesting. I haven't debugged that one down yet. I wouldn't think that should be secure. So that's the demonstration. I have sort of more complex demonstrations that we could do, too. But that's probably good for now. Oops, did it reload? Not a view.

All right. So, going back, what did we just do? One thing that I can note is I can inject addresses or inject authoritative servers, incorrect authoritative servers, anywhere on the internet, too. It would be pretty obvious if some administrator was always noticing that the addresses for everything was just the one machine in the middle box. They'd probably notice that eventually, just looking at traffic passing their network.

But I can actually inject a different man in the middle or machine in the middle over the right-hand side and say, well, okay, .org's going to go up to the root, but ietf.org, I'm actually going to send it to another one. I can create a whole fleet of machine-in-the-middle boxes, which would make it harder to detect.

That's just an aside. So don't think that this is limited to a single machine in the middle. I can route traffic or route DNS requests to basically through anything that I want. And these are all signed.

Remember, all of these zones, icann.org, ietf.org, .org, they're all signed and I can still do this.

Well, so what? You sort of get two things out of doing this meanness. One is I get a greater visibility of DNS requests that I normally can't see. My machine in the middle was able to see requests for things inside of .org and ietf.org or anywhere in the entire tree. And I can modify some things that are normally out of view. We'll come back to that one in a minute. That one's a little bit trickier.

So let's start by talking about looking at somebody else's requests. If you haven't seen Mr. Bean, he's one of my favorites. Remember this diagram before, right? Resolvers communicate with everything over network pathways, which are the dashed lines, the dashed blue lines, whereas the hierarchy tree is on the green bubbles and the gray bubbles.

And I'll remind you, this requires a machine in the middle, it requires a parent-centric resolver, so there's a lot of caveats to it. But how can I force my vantage point in front of somebody, right? If I'm a machine in the middle in front of the IETF portion of the tree, it can only really manipulate stuff below that, right? That's all I can see.

On the other hand, if I'm a machine in the middle in front of .org, I now have the ability to send those requests for anything inside of .org, be it IETF, ICANN, whatever, through another pathway to me. And then finally, if I actually am really near a root, which is harder

to do than you'd think, I can sort of take over, control the pathways to the entire DNS ecosystem. If you have a machine in the middle, and if it's a parent-centric resolver, and there aren't other things done to mitigate these problems. And remember, this entire tree, all of the records on this tree are actually DNSSEC signed, and it still works.

So, I thought I took that slide out. No. What are the parent-centric ramifications of not checking DNSSEC, right? Well, as I said, man in the middle can get more stuff inserted into them. It's actually very difficult to detect from the client, and actually, there's a bunch of corner cases that get a little bit more interesting, depending on what parts of the packets you actually overwrite.

But for just the classic example that I showed you, it's actually very difficult to detect with the client, because if the client queries the parent-centric resolver for anything that the parent should know about, that includes name server records and things like that, it'll actually get back the right answer, even though that wasn't the answer that the resolver itself used when doing resolution.

So, a man in the middle can actually modify the A and the quad A glue. That's actually very hard to detect. It can actually modify the name server records names themselves. That's actually a little bit easier to detect, because the client will get those answers back. So, my machine in the middle code actually only modifies the glue, and it uses the original name server names. It can view transactions beyond the normal vantage point, like I said.

And then the interesting one is if there were unsigned contents lower in the tree for something that you couldn't normally see, then you can lie about that too, right? You can actually say, because you're falling into an unsigned portion of the tree, and even though you couldn't normally see that part of the tree, because I've now injected myself into the DNS resolution pathways, I can actually modify anything.

So, the fun things to do would be, I think the ones that most people would already recognize, right? If you alter MX records, you can redirect mail through attacker-controlled relays. You can modify DKIM and DMARC and actually send spam and things to a target without it knowing that you were doing it incorrectly. And you can even alter HTTPS records and downgrade the HTTP version or actually manipulate the ECH protection. I won't go into how TLS 1.3 works, but that was a big change.

But unfortunately, those records in the DNS are actually not signed. Well, sorry, they're not encrypted and not signed within the HTTP protocol. So, you only get them, it can be modified in the DNS side if the domain holding it is not. And then, of course, you can add and change glue.

So, we're coming closer to the end. So, what are some quick takeaways, right? Can a child-centric resolution solve this? Yes, because it actually checks all the data, not just the data that goes all the way to the child. Is it easily visible by the child? Can the child figure out that its upstream revolver is not behaving properly? Not

easily. There are ways to, I think, pull that off, but it's challenging. The network operator, the resolver, could probably notice it a little bit more easily if they're paying attention.

And then, as I mentioned, DNSSEC doesn't catch this. I didn't put this chain of signatures or chain of data down below, but the DS to DNS key to DS to DNS key to DNS key to the final resource record like www.ietf. That's the signature pathway. Note that nowhere in that chain of information is the name servers and the IP addresses that it's talking to if it's a parent-centric resolver. But it does get the right final answer that has to be validated because of that, assuming it's in a validated portion of the tree.

And then, what about DELEG? So, DELEG is sort of a new protocol that IETF is working on that specifies how to do name servers and glue records and things like that in a different record type. Fortunately, the DELEG records will be signed by the parent. The issue all of this comes from is that parents don't sign the NS records or the address records for the resolvers of their children, but DELEG will be signed by the parent, so this will not work when DELEG comes around and is actually deployed.

So, what mitigations can we do? We can use DNSSEG-validating channel-centric resolvers, obviously. So, that's sort of what the IETF draft is about, a draft IETF DNS revalidation. I will note that if you read the security section of that document, it basically describes the entire problem that I talked about today. So, I'm not saying anything new. It's actually talked about in a number of other

places. I'm really just trying to highlight the potential dangers of it and why you might want to do things.

You can also deploy parent-centric resolvers close to the authoritative server information so that there's less possibility for getting machine in the middle. We'll come back to that in a minute. You can deploy TLS between your clients and your resolver, and that seems sort of out of place of everything I've just said, but I'll explain that in a minute. Then, look forward to DELEG. DELEG will be parent-signed.

So, I want to talk about where you put resolvers in a network, right? You have a client. It has a network between the client and the resolver, and then the resolver is potentially really far away from the root or from other authoritative engines. Well, if you move the resolver closer to the root or closer to the authoritative engines in general, especially if you have an any-casted resolver that's very widely distributed and you peer really widely, you're unlikely to have somebody be able to be a man-in-the-middle inside of your network scenarios.

However, if you do that, you've to some extent only moved the problem, right? Because now, if the man-in-the-middle was somewhere else, they could actually modify even more because now they can modify all of your traffic. So, when you move a resolver closer to the authoritative engines, you're essentially moving it away from the client unless you have a lot of peering, which a lot of the large recursive resolvers do. That's a problem.

So, this is where I said you might want to use TLS between the client and the resolver if you're talking to a resolver which is fairly far away.

Final takeaways. Parent-subject DNSSEC validating resolvers can be subject to expanded view injection attacks. It really never came up with a good name for that line. For all zones, not just ones that they would normally be able to see, they could possibly modify unsigned zones that they normally were not in the path of, which is potentially concerning. But otherwise, parent-centric resolvers, as I've said before, they're simply faster and they can be more robust in handling errors, it turns out. I'm not going to go down that line. There's actually people in the room that can probably talk to that.

And then finally, mitigations. You can choose carefully what resolver type of software you want based on where you're going to deploy it and things like that. And if you're going to deploy parent-centric resolvers, probably do a good job of finding good places where this problem is not going to happen. And then, of course, DELEG can't come fast enough.

With that, I made it. You don't want to see the backup extra complex slides. You thought all that was complex? You should see my backup slides. Are there any questions?

DAN YORK

And we will say, too, that Kathy did put a link in the Zoom chat. If you're in there for where the slides are, they're also available off the

SCED page where you can see that. And I think Warren had a question.

WARREN KUMARI: Yeah, somewhat of a loaded question. First off, hope you feel better. Secondly, isn't a mitigation as well doing local route? Because then the attacker would need to be in front of a TLD. They can't get all of the TLDs.

WES HARDAKER: Yeah. In fact, one of my slides originally sort of said that. But you have to remember that that gets you -- it's exactly what you just said, right? It requires that the machine in the middle be near the TLDs it wants to target. So if you're doing local route, you can't get to every TLD. So that's very true, Warren. Thank you. And it actually was on the slide at one point. But if your machine in the middle is in front of a bunch of TLDs, you haven't done as much either. But it helps.

DAN YORK: Anyone in the room have questions for Wes?

WES HARDAKER: How many people did I lose? That's the real question.

DAN YORK: It looks like Duane does. Daniel?

DANIEL So if we have to choose the resolver of a software, can we check actually if the ISP we're using is a parent or a child?

WES HARDAKER You know what? I haven't thought about how to do that. You probably could if you could send up some authoritative information to see what it queries. That's a good question. I suspect that there's a way of figuring that out. But I don't know. If you're deploying software, I almost had a table of which software was parent-centric versus child-centric. But I couldn't validate that my answers are correct well enough. So I did not include it in my presentation.

DAN YORK Duane raised his hand. Are you here, Duane? Oh, there you are.

DUANE WESSELS I'm here. Hiding, sorry. Hi, Wes. Duane Wessels from Verisign. So your FAQ says that child-centric resolvers protect us here because this data is signed in the child zone. Not in the case of the root zone, right? There are no signed records for the addresses of the root name servers. So have you thought about, or do you know off the top of your head, do all of the name servers and address records in a resolution path have to be signed to be protected here, or just some of them?

WES HARDAKER

Excellent question. And you're spot on. In fact, also, so in the other document that I published in the ATF comparing where local root helps and things like that, it actually talks about this case. And it actually says if you're doing local root, if you're not doing local root, you are vulnerable to exactly what you just described, right? Because rootservers.net is not signed. There's no way to validate that all the way down the chain. I did not try and put that into a demonstration, but I believe it would work the same way, yes.

DAN YORK

Okay. We may have one time for one last question. Is there anyone else? Oh, Dave. Yep, Dale.

DAVE

Oh my goodness. Sorry about that. Everybody's here. So Duane just spoke. And interestingly enough, he is the co-chair of the DELEG working group in the ITF. And I am one of the document authors for the new DELEG protocol. One of the things I wanted to highlight is that even though it does look kind of imminent that we'll finally have a draft standard out, it will probably be a while until it gets added to the root zone, if ever.

And then also to the gTLDs, that'll probably be a long time coming. Where the technology is going to prove itself, since we are fittingly in the ccNSO room, is much more likely to be in the ccTLDs to get that operational deployment experience. So while I am very

positive on the DELEG technology, realistically, don't expect it to be securing these delegation chains soon.

DAN YORK

Yeah, I think we had a session here many years ago where we were talking about the lag time around deploying new protocols within the DNS infrastructure that can be measured in years to decades, really. Yes, Peter. Yeah. One last one here.

PETER THOMASSEN

Yeah, it's just a very small comment. So what we're dealing here with is essentially a privacy issue, because the machine in the middle can observe more queries than they should be able to. Now, even if this gets fixed with DELEG, or even if everybody uses child-centric resolvers, then there are still positions where certain queries can be observed, right? So I think to actually address the privacy issue fully, it's necessary to deploy encrypted transport for DNS queries in as many parts of the infrastructure as possible. And until we do that, the solution will always be incomplete. So I'd suggest to focus on that.

WES HARDAKER

So you're not wrong at the same time. You're right that if everything was DNSSEC signed, you wouldn't be able to inject yourself in other places too. And I think one of the biggest issues is that a lot of the large domains out there actually don't use signed data at all, which means as long as you can get your machine in the

middle position through these careful hacky ways, you can do the example things that I have on the list. And this list is hardly complete, right? You can modify MX records and inject a machine in the middle inside of the MX chain and things like that too. So TLS would help as well. We're actually all getting to the second presentation I was going to give later, so maybe I will do it.

DAN YORK Well, yeah, we gave more time to this one since you were here on the show. And I'm going to give the final word to Warren.

WARREN KUMARI Yeah, just a very quick update to what Peter said. So it's a privacy issue, but it's also potentially a censorship issue. If you can put yourself on path for all requests to a specific TLD or stuff under it, you could also censor those answers because the queries are coming back through you.

WES HARDAKER At one point, you can do more targeted attacks too, right? You don't necessarily want to see the whole tree, but if you want to target this one particular zone in the tree, you could let the rest of the records sideways from that be unmodified.

DAN YORK All right, well, everyone, let's give a round of applause to Wes.

WES HARDAKER Thank you very much.

DAN YORK I do hope you feel better and get some rest there. And thank you for presenting and also for sharing or for not sharing the germs with us. So thank you very much for that.

WES HARDAKER I'm pretty sure it's food poisoning. I will likely be there tomorrow. I'm feeling a lot better. So if you want to talk to me, please come talk to me. I should not be contagious.

DAN YORK All right, well, that's good. Should not be. All right, well, thank you, Wes. And thank you. Also, just a note to folks, you can present remotely into this session. So if you're thinking of some topic, we're always open to people who can come in remotely as well.

So now, it was actually interesting that he was talking a lot about .org because just a different way of things. But .org is operated by PIR, Public Interest Registry. And PIR also has a segment of it called the NetBeacon Institute. And this is the DNSSEC and security workshop. So not everything we talk about here has to relate to DNSSEC.

And in this particular case, we've asked Graeme Bunton to come here and talk a little bit about what is the NetBeacon Institute. I

want to just ask a quick question. How many people here know what the NetBeacon Institute is? All right. Pretty good. Pretty good number. All right. So Graham's here to talk a little bit about what they are doing. So give it up here for Graham Buntin.

GRAEME BUNTON

Thank you, Dan. I appreciate the opportunity to speak here. Boy, that Wes presentation really if that was some Lego Technic set, we're going to look at some Duplo for a little bit. So I hope you'll bear with me. I'm going to do a little bit less of an overview of our work. I've had the opportunity to do that a few times with the SSAC and will continue to do that. But today I wanted to dive into a problem that we're working on solving in the hopes that this community finds that reasonably interesting. And I'm going to try and do it in 15 minutes or less. Let me start my timer.

Dan just queued this up, but for purposes of transparency, the NetBeacon Institute is a part of Public Interest Registry. The Institute was created by PIR to try and make the internet safer, not just within .org, but across the entire DNS and across the entire internet. We do a number of things, including offer services, but it's all not commercial. We don't sell anything. We don't have any customers. And we go about this in a number of ways, best practices, tools, resources, outreach, stuff like that.

We have, as I just said, a couple big projects. One of them is called NetBeacon Map, measurement and analytics platform. And this is really about measuring the prevalence of phishing and malware

across the DNS. The real principles behind this was that we wanted it to be credible and robust and transparent. And so we work with KOR Labs at the University of Grenoble in France to do the data collection and cleaning for us.

And then we do the data visualization component of this. And so we have this delightfully academic methodology that anybody can go look at. If you feel so inclined, in theory, you could go replicate this work. It's going to cost you a bunch of time, energy, and money, but feel free to go have at it. And I have a little diagram there of that process.

And so we've been running this now for three, four years, collecting data on abuse across the DNS. Maybe I'll stay here for just a sec. We've been doing this. We provide public reporting in just sort of large dashboard or large tables for public consumption. We provide private dashboards for registries and registrars. And we produce monthly reports with both the least abused and most abused registries and registrars, trying to encourage data-driven policy development. And that when we talk about abuse, we're grounding those conversations in real data.

But there's more data that we've got, and there's more data that we want to share, especially around things like mitigation rates by registry, by registrar, time to mitigation by registry and registrar, and really how that mitigation breaks down. How much of it is registrars suspending domains? How much of it is registries? How much of it is domain deletion versus sinkhole? How much of it is

action at the hosting layer? How do we have a real picture of what's happening in the ecosystem without real good data on that front? And so this presentation is really about how we're working towards that.

So we have that MAP project, which does a bunch of that work. We also have our NetBeacon Reporter project. NetBeacon Reporter is an abuse reporting conduit. It allows anybody to report abuse into it, and it standardizes it, it enriches it, grabs screenshots, checks Google Safe browsing, things like that. And then we report those to web hosts, all ICANN-accredited registrars, participating ccTLD and gTLD registries. And we gather feedback and we monitor those reports that go through. And a big chunk of the data is MAP data. So when we're measuring abuse across the ecosystem and we discover a domain that we think is both malicious and still live, we push it through Reporter as an abuse report to the registrar and the host.

In principle, it was, if we're going to measure you against these malicious domain names, we should tell you about them, and we should tell you about them in a way you could do something about it in a timely fashion. So this means we have two mechanisms for measuring mitigation across the DNS. We have the sort of academically robust one inside of MAP, and we have a sort of, rougher is the wrong word, but more limited version of this inside of NetBeacon Reporter.

So the KOR Labs MAP data is looking for changes at the DNS, so it's trying to query the authoritative servers. We're looking for changes at the domain name, and it's looking for changes at the content level. It does that in increasing intervals up to 12 hours, so I think it's like five minutes, 30 minutes, one hour, etc., and then every 12 hours for 30 days, and it does not halt on first mitigation, so it might see that the domain has gone into client hold, the registrar has suspended it, but it keeps running so that we can see maybe there's content changes, or I mean there's not going to be content changes after client hold, but it could be the other way. It could be the content has changed, then the domain gets suspended by the registrar, and then maybe it gets suspended later by the registry, things like that.

NetBeacon Reporter, so for a subset of MAP data, and the reports from third parties that go through NetBeacon Reporter has a more limited measure for mitigation. It measures against the EPP status, so client hold, server hold, and looks for deletion and known sinkholes, and it does it for increasing intervals through 12 hours, and then every 12 hours for another seven days, which is about eight days total, give or take, and then it halts at first mitigation, so we don't keep going. This means we have two pretty fun, pretty interesting data sets that we're playing with right now, where we can compare the mitigation measures across these two different data sets to see where they agree and where they disagree.

So what challenges are there in this work? Well, the content and hosting level changes are super hard to attribute or even verify. We

see quite a bit of what we think is geoblocking at phishing websites, say, and so are we getting a 403, 404, or redirect because the bad guys are blocking the geography we're checking from, they're blocking the IP addresses we're scanning from specifically.

There's a lot of reasons that we may not be able to verify that that harm is still there. The timing is really interesting, and really a lot of this work is about this attribution problem, who did what, and even though we're spitting off checks at the same time, they come back at different times, and so who gets credit for the mitigation action becomes pretty difficult to say definitively sometimes.

For example, we see an awful lot of the time that a DNS query returns faster than the EPP status check, and so we'll see, I think it was something like 90% of the DNS queries that failed had a subsequent client holder server hold within about 30 seconds. Well, of course the DNS query failed, the domain has been suspended, but who gets that attribution? We also see domains go up and down very quickly that may have nothing to do with registry or registrar or even host action, it could be that the bad guys have finished what they're doing, they're rotating domains really quickly.

We run into rate limits, registrars reasonably don't love it when we're hitting their RDAP servers really hard. There's lots of ambiguity in this data did the, if the domain is deleted, who did that, who changed the name servers, there's a lot of scenarios here, there's really edge cases for just about everything that you can

think of, which makes this attribution pretty difficult. Boy, we want to get it right, we want to be able to stand up in front of this community as we get closer to publishing more data and be able to say with some authority, like, who's doing what?

This is an example from our website of the existing mitigation measures that we provide, green is mitigated, totally un-attributed, so inside that green bar is going to be DNS level measures, WHOIS, which is really like EPP status, content level changes, totally could be bad guys, could be registries, registrars, could be anybody in that sort of internet infrastructure chain, and we really want to provide more insight into what's happening there.

We also have mitigation speeds, we measure this across all of the data that we've got inside of map and reporter, and so now we want to know and could know for some of this data, is it the registrars that are acting quickly, is it registries stepping in first who's doing that, if we've got the attribution, who's doing it the quickest, and things like that are pretty important. Generally, you can see here that for most of the data that we're measuring, and this is from our MAP data, to be clear, we can see that most of the mitigation we're seeing is happening inside of 48 hours.

So what we want to be able to provide once we get through working through those challenges is going to be way more insight on a chart like this. So this is mitigation rate by registrar, but included in that is going to be server hold, which will be where the registry is taking action, that could be where the bad guys finish doing bad things.

We just see that those domain names or those harms came offline some point in the 30 days that we've been measuring. I'd point out that the registrar 9 here, who's pretty anomalous, actually has a large amount of compromised websites. For whatever reason, the market that they serve is bigger and older, and so they have more probably aged WordPress that is getting compromised, and so where abuse is compromised, we see much lower mitigation rates.

This is kind of a fun one, and it requires a bit of explanation, so I'm going to dwell here for just a minute. This is mitigation rates measured from NetBeacon Reporter. So this is just client hold, server hold, sync hold, deletion, no content level mitigation measure, and it's from inside our reporter project.

So there is a significant lift to these mitigation rates when I check these domains against the map data, where we see mitigation happen later than the eight days that we monitor for or at the content layer or hosting layer, but this is the people who are reporting through NetBeacon Reporter, the different organizations that are using this service to try and disrupt abuse, and there is a real substantial amount of variety in the quality of the reports that they send, and their consequenting mitigation rate.

So I think it's the sixth one in there is a dude from Canada who tries to disrupt phishing as a hobby. He's built himself this cool little platform that goes and finds phishing domain names, grabs a screenshot, and automatically reports it to us, and he sees somewhere between an 85 and 90 percent mitigation rate. Some

of those other bars are cybersecurity companies, consumer protection, brand protection, and they vary a lot.

I'll remind people, again, their true mitigation rate is higher if you're looking at those other factors, including content, but there is a lot of variety, and so one of the things that we're working on doing is building better processes for working with those reporters to say, hey, your mitigation rates are generally pretty low, and they're pretty low because you're providing low-quality reports, which could be your descriptions are too long. It could be your screenshots are bad or missing. There's lots of reasons like that where people aren't great at reporting, and we can try and help them get better, and that's one of the benefits of running this conduit and getting more reports through it is that we can try and lift up everybody to see more mitigation.

So what's next for us? Well, this is a screenshot from one of our monthly reports that we provide. You can see that we've got your observed malicious per 100,000 and the number of malicious gTLD names, so this is the most abused registrar table. The number of months is a consistency requirement, so you can see there's a redaction here. We redact registrars that weren't in the top 10 most abused for four of the previous six months, so you need to be in there for more than four of the previous six months, and we did that because we didn't want anybody to, and we do see this.

We see this in our data that registrars get impacted. Sometimes for just a month or two, they get targeted, and then they get better.

They figure out what that problem was and that abuse goes away, and so we were really trying to, if we're going to name you in this chart of most abused registrar, we want to make sure you're consistently up there, not just a one-off. But we don't, in these charts, share mitigation rates by registrar, and we don't publish the median times.

And so that's really what we're working towards doing is reducing those redactions, adding mitigation rates and mitigation times, but we want to be really confident in those mitigation attributions so that when we say this registrar mitigated this percentage of abuse, we can break that down into how much of that was client hold, how much of it was server hold, how much of it was deletion, how much was happening elsewhere at the content level.

We also have, as we build these tables, a lot of exclusions because the data is quite noisy. You know, if you don't have any domains under management, you don't have any abuse, should we put you in the top 10 table for least abused registrars? You know, it ends up with thousands of records in a table that isn't providing this community, say, with the information it needs., so we need to, as we get closer into publishing this data, homogenize those exclusions across our least abused and most abused lists like that.

And ultimately, the goal is just to get to a full table so that we're publishing for all TLDs and all registrars all of this data plus mitigation rates and mitigation times, but we really need to get through this process of cleaning up and solidifying the attribution

of our mitigation data. And we're really hoping to have this, oh boy, I probably shouldn't say publicly how long it's going to take us. It's going to take us a little bit. It's probably another six months or so would be my guess because it's really difficult work, but I hope it will be of significant value.

We had a couple of final thoughts. Measuring abuse is really hard. Doing this work is complicated and time-consuming. Measuring this mitigation is substantially difficult. Getting around those geoblocks is nontrivial, and then accurately attributing that mitigation is harder still, and like always, we're working towards increased transparency in this data, and so I'll stop there. Thank you for the opportunity. Happy to take some questions.

DAN YORK All right. Any questions for Graeme? Come on. You can't just let Graeme off the hook that easily. I see you laughing, Jeff. Anyone? All right. Well, Graeme is around. No?

All right. Let's please give Graeme a round of applause. And you can go to netbeacon.org to see more about all of this that was there. Graeme, there was somebody in the chat did mention that the link to the map analytics link is returning a 404, so I don't know what that was. Anyway, with that, Kathy, I just need to share one thing here.

KATHY SCHNITT Scott, can you make Dan your co-host?

DAN YORK

Details. I just want to say thank you to all of our presenters here. Peter, for coming and providing that statistics. And Peter, there was a note actually. Jeff actually left you a question in the chat as well that you can take a look at there, and Peter is also noting something here and says, look at superdns.nl to look at testing for the child-parent-centric resolver.

Warren, do you want to just give a quick heads up about what you're going to be talking about in the lightning talks? We have like five minutes. Okay. Go for it. I want to thank Warren for being on standby if Wes was not able to present. So, Warren is going to be talking about something in the lightning talks. If you're not aware of that, the Security and Stability Advisory Committee, otherwise known as SSAC, has lightning talks twice, once on Wednesday and once on Thursday in one of those sessions, and I don't know which one.. We think Thursday, but Warren will be presenting on this, and here we go.

WARREN KUMARI

So, yeah, this will just be a quick overview. Basically, we can no longer publish things like that in the DNS because of privacy reasons, so instead I'm proposing something called blinded unique identifiers, which will allow people to do associated domain checks across registrars in a privacy-preserving manner. So, if you're interested, come to the SSAC lightning talks on, I believe, Thursday.

DAN YORK

Thank you, Warren. All right, and we get a thumbs up from the back there saying it is Thursday. Okay. So, I will just say I want to give a word of thanks to the people who are on this slide. They are all the folks who've helped bring together this DNSSEC and security workshop that happens at each of the ICANN meetings. So, these are the folks who are out there looking at what sessions are available.

You'll see a call from proposals that we put out for every session. We tend to circulate it on a lot of various different DNSSEC lists, put it on different places, and we'll be looking for presentations for ICANN87. So, if you've got ideas that either want to present remotely or in person, if they're new research like what Wes was showing or what Peter was showing, if you've got something else adjacent, we've had presentations before on routing security.

We've had the people from MANRS come here and present a number of times. We've had some other different presentations. If it's related to ICANN-centric kind of security topics that, in some way, they do not have to be about DNSSEC. It can be something related to security in that kind of space.

So, are there any final questions for any of our panelists? Seeing none, I will give you four minutes of conversation. So, thank you, everyone.

KATHY SCHNITT And thank you, Dan. It was great to have you back. Scott, please stop the recording.

[END OF TRANSCRIPTION]