ICANN85 Mumbai | CF – SSAC Lightning Talks (1 of 2)
Saturday, March 7, 2026 – 13:15 to 14:30 IST

KATHY SCHNITT

Hello and welcome to the SSAC Lightning Talk Session. My name is Kathy and I'm the participation manager 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.

Regarding participation today, this session is primarily designed for the internal work and discussion of SSAC members. To join the speaking queue, SSAC members must be logged into zoom and use the raise hand feature. When called upon, please state your name for the record. For observers, please note that comments or questions in the zoom chat will not be read aloud. However, the chair of the session may invite questions from the audience at their discretion and time permitting. I will now hand the floor over to Barry Leiba.

BARRY LEIBA

Hi, everybody. Those of you who have heard me do this before know that this is my favorite of all the SSAC sessions. I love the lightning talks. And we have four good ones today. So let's get right into it. First, John

Levine is going to tell us all we need to know about DELEG, some new work the IETF is doing. John.

JOHN LEVINE

Thank you. So, Kathy, you're driving. So, next slide please. This is work that has sort of percolated up over the past year or two, and the problem, it's not so much that anything is broken but DNS delegation, the way you link a higher level to a lower-level zone file is complicated and fragile.

The way it currently works makes it pretty much impossible to do secure queries like DNS over TLS or DNS over QUIC, and it's complicated and messy. So, here on the right, we have the DNS camel crying in the desert and we're trying to make him a little happier.

Next, please. So here's how delegation works now and in all these examples, the yellow stuff are DNS records that are signed, and the white ones are DNS records that aren't signed. So, we have here, my domain name example.com spelled with the digit one, and what's up at the top is what you might find in the.com zone file.

So there's a DS record that links to the DNS key that's going to be in the lower-level zone file and then there's an NS record that says, okay, the name server for this zone is ns1.example.com, and because the name is underneath the point that we're delegating, we also have a glue record here that says, as a hint, here's an IP address for ns.example.com, and then in the lower-level zone file, basically, everything at the top is signed, the SOA is signed. This one happens to have two keys; there's a

key signing key and a zone signing key, and the actual copy of the NS and the A.

So, next please. Something that is I think poorly understood is that the NS and A in the upper level are just hints. They are not definitive and they might be wrong. So, the only things that have to be corrected, the DS and the upper level have to have the correct hash for the DNS key in the lower-level. So, next please.

And so, the NS records in the even the glue in the lower-level don't have to be the same as the one in the upper level. And in fact, for those of us who manage lots of DNS zones, they frequently don't match, which I say usually works except when it doesn't. So, this is a mess.

Next, please. Yes, surely, we can do better. So, at this point, it is like DNSSEC proves that everything gives you a chain of signatures that tells you that the data is correct, but it doesn't tell you anything definitive about where to look for it.

So, next please. So, the IETF is in the process of inventing a new replacement for all this stuff called DELEG, and we have for DELEG, whereas the old thing is sort of bottom up where you discover the name servers in the lower-level zones, DELEG is now strictly top down.

So, here in the upper-level zone, we still have the same DS record, but now we could have some DELEG records and you would not typically have all three of these, but the DELEG record is going to say, here's the IP addresses of the name servers for the lower-level zones. Here's the V6 addresses for the name server for the lower-level zones, or here is the

name of the Name Server for the lower-level zone. And the lower-level zone still has the SOA and the DNS keys, but it doesn't have the NS anymore because the definitive information about where the name server is, is in the upper level.

Next, please. So yeah, what I just said is, these are now authoritative. It says, like, if you want to find example.com, here's the IP addresses. And here's the name of the server. And those are the only places you can look. And these are DNSSEC signed. So, the lower-level zone has no opportunity to change this.

Next, please. One thing this makes go away is the glue records. The hint about the A or the quad A and the upper-level zone, you don't get that anymore. If you want to point at a name server where the name would be in the lower-level zone or that has no name at all, you can simply say, here's the IP address. So the DELEG server here goes directly from the upper level to the lower-level by IP address with no names at all. You can use names, but the names have to be somewhere else that you can look up without having to follow this chain.

Next, please. Another thing that ads justifiably valid complication is another record type called DELEGPARAM, which has the same syntax as DELEG, but whereas DELEG says this is his own cut and when you get to this record, you need to go find a lower-level somewhere else. DELEGPARAM is a regular record and the point of DELEGPARAM is, the example here, is you can say, include DELEGPARAM pointing at something else.

And the point of DELEGPARAM is that if you are a – I mean, for example, I run a smallish name server, but I have 300 zones, you know, each zone has the same NS records and it would be very nice if I could clean this up and say, in the 300 delegations, we simply haven't included that all point at a DELEGPARAM that then has the definitive information for the name servers I'm going to use.

So if I want to change the names of my name servers or locations, the name of the name servers, I can just change the DELEGPARAM and everything else follows along. So, it is CNAME like redirection. You can actually have a chain of DELEGPARAM, include can actually point to a CNAME which points to a DELEGPARAM, which I don't think is a good idea, but you can do it anyway. So, this makes the administration for large name servers somewhat simpler.

Next, please. So, how do we get there from here? The semantics are different; for the foreseeable future, you need to do both. You still need to have the NS records and you still need to have the DELEG records. And if you're going to have glue, you still need to have the glue. And the way you can tell which one to use is that when a DNS cache makes a query to an authoritative server, we've invented a new status bit called DE for DELEG enabled.

So, if you get a query with a DE bit, then if there are DELEG records, you use them. Otherwise, you don't and you return the NS. Actually, this is wrong by the way, the NS is here, shouldn't be signed. Whereas if the query does not have the DE bit, then you do it the old way. And the Internet being the Internet, we expect these things to coexist for a long

time and the DNS Op Working Group is in the midst of a fairly heated argument about how exactly is this coexistence supposed to work.

Next, please. This is pretty arcane. There is a new bit that you put in your DNS key that says that this server understands DELEG so that if you get a referral from the server and it doesn't have a DELEG, then you need to go back and check the signature of the upper level and make sure that there is in fact no DELEG there and that hasn't simply been stripped by a malicious middle box.

For all practical purposes, your DNS cache will do this for you, but it does mean that when you're moving up to DELEG, you're going to have to add the extra status bit to your DNS key, which then changes the check sum of the DNS key, which then changes the key tag of the DNS key, so you'll have to do a key rotation. However, if you have both a zone signing key and the key signing key, you can put this bit in either key, so you only have to rotate the KSK, which is easier than rotating the ZSK.

And I think I have one more slide. Yeah, so open issues. So, like if you have a zone cut that only has DELEG and you get a query that doesn't have a DE, what do you do? And I have opinions, but I'll simply say that at the moment this is unresolved. I mean, there's some people think that it should try to simulate NS records, which I don't think will work very well. Or there's others that you should simply say, you should give a server fail, or you should say NX domain, or you should do something, but we'll work that out.

People have started to implement this. I think even some people in this room may have started to implement this and it seems to work. But

since this is very early days and this is a pretty significant change to the DNS, we need more experiments to make sure we believe it works. And barring surprises, we hope we can publish the first [inaudible – 00:10:55] later this year.

And one more slide. The security model has now fundamentally change. So, the DELEG lets you say exactly which servers have your DNS data. The DELEG syntax of the key value stuff is extensible, so that we assume that once this works, we can add some more tags and say, here's the server you should use and you should query it using DoT or you should query it using DoQ.

So, this is a significant change because this means that you can now have a chain of queries. They're both DNSSEC side and that are encrypted so that third parties cannot have a lot more difficulty telling what DNS you're going to use. If you're using DoT or DoQ, the server will present an SSL certificate. So, there's some complication in making sure that the SSL certificate actually matches the thing that you're pointing at. ACME, which lets you encrypt, has recently added a hack so that they will now sign SSL certificates for IP addresses, which is pretty important if you're going to do DELEG IP addresses.

And in the very, very long run, if you actually have a sign chain of delegations where it's encrypted, the SSL certificates are all valid. Then at some point in the distant future and people very how far distant is, but you know we're talking like a decade, it might be that if you know the server is that trustworthy, you might be able to just believe the data inside it without necessarily having to check for DNSSEC.

So, in a sense, at the moment this provides path authentication in addition to DNSSEC data authentication, but it's not out of the question that in the long run, that will end up using path authentication instead. So, I mean, it is a change. It's potentially going to affect everybody who uses the DNS. At no point we ever have to upgrade to DELEG. If NS works for you, it'll continue to work forever. The reason you would want to use DELEG is if you want to do DoT or DoQ to encrypt your authoritative server queries to make them snoop resistant, which I think is a good reason. I think that's the last slide. One more? Yeah. Do I have a minute or two for questions?

BARRY LEIBA We do. So, three things quickly. One is, if you have questions, please raise your hand in zoom. We're not going to take questions from raising hand in the room or anything like that. Two, as Kathy said before, priority given to SSAC members, but I will take questions from anybody if there's time available. And three is, we will likely have some time at the end for additional questions, if we don't have time right now. So, questions to john.

JOHN LEVINE I've stunned people.

RAM MOHAN John, so just to understand, there's going to be both legacy NS and DELEG records in parallel for the foreseeable future?

JOHN LEVINE That is what I expect. I mean, some people think you can just put in DELEG and you can fake the NS, but I don't think that will work. I think you'll actually need both.

RAM MOHAN Just sounds like if there is no path to deprecation for NS, feels like it's going to be a messy future.

JOHN LEVINE I mean, the path of deprecation is, sort of depends how important people feel that encrypting their queries are. I mean, if you want your queries to be encrypted, then that's a pretty strong motivation to use DELEG and it also really depends how good your automation is.

Like on my little system, the NS records are all generated from macros. So, all I need to do is like add a DELEG line or two to the macros and poof, it'll all work, but I realized that different providers have different levels of automation and it can be anywhere from trivial on my system to manually editing 1000 files with VI, which is not so good.

DANIELLE RUTHERFORD All right. We have time for one more question, and Hadia?

HADIA ELMINIAWI Thank you, John, for this presentation. And as you mentioned, this work hasn't been published yet, but it's currently being tested, and my

question is, are you looking for other people to test this with you, or can maybe deploy it and implement it and join your testing team?

JOHN LEVINE

This is happening in the DNS Op Working Group in the IETF, and the test software at this point, as far as I can tell, is mostly being written by the same people who write, you know, like unbound and bind in the software we're using now. I mean, the more testing, the better, but it is pretty clear from the discussions we're making is that, unless you already understand how to write a name server, writing a name server from scratch with DELEG is a ridiculous task, and so, what you'd do is you would start with a name server that already works the way it works now, and add new code to handle DELEG, and I think the major name servers all have people doing that stuff.

So, it would be nice to have more people testing it, but I think, practically speaking, until there's actually software that they've deployed that you can install, there's probably nothing -- unless you happen to be one of the handful of name server developers, I don't think there's anything to do yet.

But again, DNS Op is open to everyone and the mailing lists are all archived; and although the RFCs have not been published, there are plenty of drafts that are published. I mean, this talk is extracted from the DELEG drafts that say what the current status of the design is.

Oh, I'm reminded that this was spun off of DNS Op, it has its own Working Group called DELEG, whereas the mailing list is called DD.

BARRY LEIBA Okay, thanks, John. Now, we've had some Lightning Talks before about this really interesting phenomenon called bitflips that I think is very cool, and Peter Thomassen is going to talk about some recent measurements thereof.

PETER THOMASSEN Hello, I'm Peter Thomassen, I will talk about Bitflip effects that can happen when you consider the domain root-servers.net which is an important domain because that's where the root servers live. It's 20 minutes. I don't know, it's a bit of content. So, we can also discuss questions after the session.

Very quickly, introduction to Bitflips. So, computer hardware like random access memory is in general subject to issues, for example, corruption of Bits that can happen through environmental stuff or cosmic rays and the probability is small, but there's a lot of memory on the internet, so maybe it is significant.

The nice thing in terms of exploiting it is that you don't need to do any reverse engineering or some other sort of manipulation. It just happens. And if you're lucky enough, you can exploit it. There're various places where it can occur. And it destroys the integrity of the data that is being stored. So, the security model essentially is that, if you can do some nice attack through exploiting corrupted integrity, then a Bitflip might help you.

Mitigations are error detection and corruption. So, it's so checksum sorry. Mitigations error detection and correction. So, things like error correcting codes and memory, for example, and that's not very widely deployed, but it exists. So, you can actually fix your hardware to prevent this.

That was the introduction and the inspiration for the experiment was from Warren's experiment from last year in June, where he showed how certain Bitflips and commonly used domain names like microsoft.com can be exploited to receive traffic on things like microsont.com and he was able to figure out some certain cloud API endpoint names and extract cookies and SNI names from TLS handshakes, and so, that was all pretty interesting. And there's a link here in this talk if you want to download it later and watch Warren's talk that was the inspiration for this experiment.

So far, so good. Now let's look at the DNS and the root server system. I don't think this has been done before. So, a quick introduction to how it works. You don't need to read all of this, but there is an RFC that describes what priming queries are. A priming query is essentially a way for a resolver to update its knowledge of what the root server names and IP addresses are, and so that's because as it's written here in boldface, root server names and addresses can change. And so, resolvers need to use priming to get an accurate list at a given time.

The way that works is that you send a query of type NS for the root name, which is the empty label. And so, you can do that when you start the resolver or when your knowledge has expired by looking at TTS, for

example. And of course, an on-path attacker can fake that stuff. And maybe someone who receives a bit flipped query can also fake that stuff. So, let's see what happens.

In practice, a priming query looks like this. So, this is a Dig command that is for Q type NS for the root domain and to get a bunch of the root host names and the glue which I've omitted here. Initially, when you are starting your resolver, you have a hard coded list of root name server host names and their associated IP addresses from what's called the hints file. That's usually hard coded and resolvers might update it at runtime by issuing a priming query.

So, the priming query RFC also says, an attacker controlling a rogue root name server also has complete control over all unsigned delegations and over the entire domain namespace in the case of non-DNSSEC validating resolvers. So that's what we're trying to exploit here. And actually, the root mainly carries referrals and as John just explained, having them may be authenticated and signed with DELEG in the future might be useful.

All right, so let's move on to the woot servers experiment. All right, so here's the setup, we registered 48 bitflip variants of root-servers.net. It turns out there's no delegated .net Bitflip variant in the root zone. So we didn't bitflip .net, but there's 56 I think variations of root servers that you can look at.

Eight of those were already taken. I think bootservers.net is already taken. We registered the 47 that were available in July last year and one other became available a bit later, so we registered that too.

Interestingly, of the 56 variants, 36 are where a bit gets cleared and 20 are where a bit gets set. The reason for that is that lowercase letters by clearing one bit bring you to a number, a digit, whereas that doesn't work in the other direction. And that is why there is more clearing opportunities than setting opportunities.

Those Bitflip variants of root-servers.net domains were delegated to our A and B woot-servers.net infrastructure, which hosts our fake root-servers.net domains, and then we collected traffic of the queries that arrived there and we also in a second step that I'll talk about later started serving the actual root zone, so let's look into what the data contains.

The measurement was done from July 2025 till mid-January, after which we did a first analysis and later some adjustments to the experimental setup, but I'll also get to that. There were 16 gigabytes of traffic for the domain names I just mentioned. So, that's not actually root traffic, but that's just traffic for ABC and so on, root-servers.net with a Bitflip variations. And then about the fake root traffic, actually there is 220 gigabytes of traffic that needed a bit of resources to process. And my alma mater Rutgers University was kind enough to provide that.

So, let's first look at the queries we got for the single letter .Bitflip domain names. So, we have two virtual machines. One hosts the a.woot-servers.net domain. The other hosts the b.woot-servers.net

domain. They are in different IP address networks and so on and so forth, as you're supposed to do it. And so, here's what we got.

This is July until January. There were a few thousand A and AAAA queries every day. This is for a.woot-servers.net. This is for B. So, I flip back and forth. That's essentially the same thing. So, this was telling us that there are actually such queries, although we don't know necessarily whether that some stuff that people just throw at our servers or whether it's actually Bitflips, although it would be unlikely to come up with like a weird Bitflip that, for example, would replace some letter by a digit and then ask that randomly, why would you do that, but okay.

We also looked at the number of query sources, which is unique IP addresses. And so, essentially every day that was between 1000 and three or 4000 and there's more IPv4 than IPv6 as usual. And yeah, so that's not very interesting, except that it shows that traffic is actually happening.

And this is a diagram that is not very, let's say, you know, the colors are like the holy festival. What's really the most interesting part is that the biggest slice here of this cake corresponds to one particular variation, which is rcotservers.net. It turns out, if you look on the right-hand side that the O and the K differ in the third bit from the right, both in the uppercase and lowercase variation, obviously.

And then the second winning variant is the root-servers.net, where the last E is replaced by a U. That's the orange slice here. And that is a different bit that gets replaced. All the other ones essentially got around

34,000 queries each in the observation time window and the variation was between 2000 and about 100,000 queries per name.

There were three incidences of rcotservers.net, which actually has two bitflips, which is, I think, interesting because it seems quite unlikely to happen. I don't know, servers, R and V, I don't know which bit that is. It might also be the third bit. So, maybe there is some hardware fault in that machine that was asking those questions. Note also that we didn't register that name, right?

So, we only registered names with one Bitflip and then we still received a question for two-Bitflips. So, apparently there's different points in time where Bitflips can happen, for example, after doing the first resolution before sending the next query or something. All right, so this is the same thing for quad A queries. The rcot thing won again. There's another variation on the second rank, whatever. There were some other prefixes that were queried.

So, the single letter ones here are shown on the left-hand side. You can ignore the four right columns for the moment. So, in the middle, the 13 single digit or single letter names are shown. On the left-hand side, query statistics are for A.woodservers.net and in the middle of the table for b.woot-servers.net. We have A queries and quad A queries and you can see that most queries are for A and then the rest of them is essentially more or less randomly distributed with like some sort of around the average of, I don't know, a few 10,000.

What's interesting is that there is a bunch of other prefixes that you would not expect. So, I only looked at single character ones, for

example, a backtick and a closing parenthesis and stuff like that. We already found that O and K are a Bitflip. We saw that on a previous slide, K is actually a root server letter and O isn't. So, some of them here show up, 224. I don't know why some are so frequent, like for example, the backtick has 40,000. I don't know, it's just a finding.

All right. So, the second step of this experiment is to serve the real root zone on those fake root servers. The data set for that has around 150 million queries, 75% of which are any queries. So, an any query is a query type where you are asking the name server to send whatever record it has for the name. For some reason, Europa.eu and some other names are very common. 25% almost are TXT queries for names like Cisco.com, Atlassian.com, GWU.edu.

If you look for the TXT records there, you will see that they are actually very large. It's several kilobytes. So, these queries look like bots or something that send TXT queries to random IP addresses to figure out if they can do an amplification attack there by spoofing the source address. I mean, it's a hypothesis, but this is how it usually works.

And so, we removed those from the data set. It's actually 99.9%. I guess that's how much of the DNS works. And after pruning and removing other invalid queries like for the chaos class, around 60,000 queries remains. And that's, yeah, it's hard to tell whether that really means those are actual queries or those are random queries. So, what we need to look into is whether there are actual resolution cascades where we can follow that somebody first queried a bitflip variant of

rootservice.net with a single letter in front and then later use a fake infrastructure, for example, to look up a TLD.

And that's not easily visible from the experimental setup that I've explained so far. So, we made some adjustments, but before I get into that, let's look at an interesting set of queries from a European mobile ISP.

Those are from various months, but they are from the same subnet, and they sort of all relate to car stuff. So, for example, in July, there was a query for some Cloudflare hosted domain by Garmin.com. Garmin is a navigation hardware provider. And, yeah, so this query somehow arrived on our systems. Then two weeks later, there was one for volvocars.com. We'll see that again later. This is at a European ISP.

In September, there were queries for some AWS things. This name that is queried here is actually a CNAME target for some other Volvo name. So, this seems to be a follow-up query. What's interesting is that this name has additional Bitflips in it. For example, the S at EU-SEST is probably supposed to be EU-WEST, because that's how the Amazon naming structure works. And this EU-WEST here has the hyphen replaced by a closing parenthesis. And if you look in the bit structure of those octets, you will see that those things are flipped on the third bit on the right, just like the O and K that we saw earlier for the rcot servers.

It actually turns out the rcot queries were from the same European mobile ISP. Then some of those car-related domains are hosted at Amazon, so there's AWS DNS involved. And so, some of their hostnames

are .co.uk. There's also .ck.uk query, and again O and K are a part in binary representation by the third bit.

So, apparently this ISP has one or more devices that are affected by such hardware fault. It doesn't seem random, so it's not a cosmic ray or something apparently, but a structural hardware problem. It's hard to tell whether this is customer equipment or whether this is a resolver. I don't know.

Okay, so let's improve the experimental setup. Initially, we only responded to A and quad A queries for the single-letter .Bitfliproot-servers.net domains. We did not respond to NS and DNS key queries and such things, because the setup was very simple. We fixed that. We gave proper answers for all queries now, including no data and so on and so forth, but we didn't drop anything anymore.

The second improvement was that whenever a bitflip variant of the root server was asked, we included the response to that, but also the real corresponding hostname. For example, if you ask for a.root-se2-verse.net, we would include a response for that, but also include a response for the regular A hostname, but with a different IP address on a different VM.

So, we have p and o.root-servers.net, which are the two VMs. And then depending on with which destination address subsequent queries arrive in our packet captures, we can tell whether the one or the other answer was used. So, this is interesting, because it allows distinguishing failure

modes of where in the process the problem happened. We do the same thing for our pv6 queries.

Then, finally, we thought maybe it's interesting to also tailor the priming response, because let's say some resolver gets hooked up to our nameserver infrastructure, and then the first thing it does might be issuing a priming response. If we serve the real priming response, then the effect is undone, because it updates its knowledge with the actual priming response from the root zone.

So, we tailored that a bit, and without touching any other root zone responses, we answered the priming queries with our own two nameservers, but again, four other IP addresses, 2IPv4 and 2IPv6, and those are not the same as on the previous slide, which you get when you resolve the single letter names manually.

So, we can distinguish whether arriving queries actually are from a priming response or from looking up a single letter name. And the nice combination, or the interesting combination, is where you first look up the single letter name, and then later the resolver does a priming query and switches to the other IP address. Then it looks like the resolver has latched onto the hostnames in glue.

So, let's see if that works. There are some interesting queries, for example, related to the Ford Transportation Mobility Cloud. This is not blaming Ford, because they don't operate the resolvers, they just use this domain that is being queried. And first, priming responses were sent to the IP addresses of, for example, a.toTheBitFlipName, and then we returned other IP addresses for the fake root service in the priming

response, and then subsequent queries were sent there. So, it seems like that resolver software switched.

It's not clear why so many queries for the same thing are sent by the same IP address in a few minutes. It cannot be DNSSEC, for example, because we didn't touch that. No failures there. And the last thing here is, there's also USMobileISP that, again, did some priming queries. Here, the Bitflips are in the second bit from A to C, for example, com.aom, the thing that is queried here is actually a CNAME for some CloudFront thing, and I'm not showing that here, but there's a lot of CloudFront Bitflip queries that follow.

Then G-Book is affected, which is, again, a navigation thing used by Volvo. The IP addresses used here are the ones from the priming response, so there was also a switch here. This is, again, Volvo, which has been flipped to Volto in the second bit here. And then finally, the Volvo infrastructure, volvocars.com, uses nameservers, which in turn use nameservers, which use cscudns.org, which we saw queries for.

So, it seems like those are resolution cascades. And because that involves various TLDs, the gtldservers.net questions also appear. So, these look like real resolution cascades. To sum up, there were about 10 million queries for the Bitflip, single letter names, and about 100,000 queries on the fake root name servers after pruning. It looks like there were about 10 resolution cascades observed with the improved setup, which was only in the last three weeks.

So, more might be there if we let that run for longer, or if we actually faked TLD referrals, which we haven't done, because that would be

more disruptive, but probably it would be possible to sort of escalate that. The analysis could be improved by looking at country and ASN and stuff like that. And the question that I'd be interested in from the SSAC, I mean the answer I'd be interested in is whether you think that's worth pursuing, or whether the whole thing already is mitigated because the names are not registered. Thank you.

DANIELLE RUTHERFORD All right; we've got a question from Warren.

WARREN KUMARI So, firstly, thank you very much. This is really cool. Thank you for doing it. Two sort of more observations. You saw multiple Bitflips for some names. When I did my testing, I also found that I was looking for much longer URLs. I think what happens with that is this is generally an ionization event. And so, if an ionization thing happens in one cell, it's likely to affect nearby ones as well. So, often if you get one Bitflip, you'll get the other, a bunch more.

And then the second bit was I also saw a large number of results for Volvo and some other car manufacturer. And I think a hypothesis for that is these types of devices often are in things like that they sit in the sun for a long time. And you're more likely to get Bitflips in memory chips that are sort of unhappy slash overheated. I don't know how we test that hypothesis more, but it does seem potentially related.

PETER THOMASSEN Yeah, it does. So, about the ionization, if you look on this slide, there is queries from July, September, and October. And they all relate to the flips in the third Bit. So, I'm not sure why the ionization random event would always relate to the same Bit. It seems more structural to me, but I don't know.

BARRY LEIBA Okay. Thank you, Peter. Now we got one from Nabil Benamar is going to talk to us about how prepared we are or are not. What? Okay. Ram.

RAM MOHAN Sorry, Peter. On the previous one, you had asked, is it worth pursuing? I think from the SSAC point of view, it's a phenomenon that we're kind of observing. And we now have two reports on it. To pursue then would mean do we start up a work party on this? Is there something to study and is there something to recommend? And I don't know that it's risen to that level yet.

PETER THOMASSEN Yeah, that's not what I meant. What I meant is whether you think it would be useful to continue collecting data and see what's in there and stuff like that.

BARRY LEIBA Okay. So, Nabil will talk to us about AI-driven cyber threats. Go ahead.

NABIL BENAMAR

Thank you, everybody. Thank you, everyone. So, Assalamu alaykum. My name is Nabil Benamar. I'm an SSAC member. And I'd like to talk today about AI-driven cyber threats and the preparedness level that we have. So, how prepared are we really?

Next slide, please. So, as you can see from the slide, as of February 25th this year, the world's most popular penetration testing platform, which is Kali Linux, officially integrated Claude AI. So, what does this mean, actually? So, a tester can now type plain English and have AI autonomously run full attack chain, reconnaissance to exploitation. In plain English. No syntax, no memorization, no needed expertise in the field. So what used to take three hours now takes less than 15 minutes. And without really big expertise in the field.

Other reports, like the CrowdStrike 26 Global Threat Report published this year, documents threats actors already using this same protocol developed by Anthropic, which is Claude Code MCP, the model context protocol for offensive operations before the Kali announcement was even made. So the question this talk is going to answer is not whether AI has changed the landscape or the threat landscape, something that we are more investigating in the DNS Abuse and AI work party. But yes, the AI has already done this. And the question is whether our defenses have changed at the same speed.

Next slide, please. So, based on a survey published also by Accenture, 36% of technology leaders acknowledge that AI is outpacing their security capabilities. Yet, a staggering 90% of companies lack the maturity to counter today's AI-enabled threats. So, the survey covered

more than 2,000 executives from organizations all over the world, 17 countries or more, and conducted online from the end of October to December 24. So, it's a bit of a comprehensive survey.

Next slide, please. Also from this survey, we can see that only 34% of organizations have metered cyber strategy. The vast majority remain exposed and are prepared and at risk of failing behind as AI-powered threats accelerate. Another study now from Cisco. So, while there have been significant advancements in AI, this report that has been published in 2005 shows that the AI deployment in cyber security defenses appears to have stalled.

Companies are still grappling with concerns around trust, effectiveness, and integration. Many security leaders hesitate nowadays to fully embrace AI-driven defenses due to different things, like, for example, due to technical feasibility issues. And only 48% of these companies believe employees understand how malicious actors use AI.

This report from Cisco goes beyond this and concludes that only 7% of companies have reached the major stage specifically for AI fortification, something that we can dip into later. Now another report published by the World Economic Forum shows that while 66% of organizations expect AI to have the most significant impact on cyber security in this year to come, only 37% reports have been processed in place to assess the security of AI tools before deployment.

So, this is a big issue. Whenever we want to deploy any AI-based solution, we need to test it first, to assess it first, before we can deploy it. And for this, you need expertise in your company, enterprise,

organization to be able to handle this. So, this reveals the paradox of the gap between the recognition of AI-driven cyber security risks and the rapid implementation of AI without the necessary security safeguards to ensure cyber resilience.

There is definitely another issue related to the shortage in cyber security experts or talents. As organizations integrate more AI-driven tools and systems, they also need specialists with AI-aware security skills. So, there is a clear and balanced AI landscape. The impact is already visible, and the staffed teams face increased workload, slower response time, growing burnout. Meanwhile, attackers, by using these AI tools, are using automation and AI to increase speed, scale, and sophistication, further stretching the limited resources.

We have another issue here, which is the shadow AI. Shadow AI is not a future problem. It's happening right now. It's in your organization today. Over half of your employees are feeding sensitive data into tools you have not approved or cannot monitor. Shadow AI refers to the AI tools and applications that employees use at work without the knowledge, approval, or oversight of their IT or security teams.

So, this is part of the continuity on the shadow AI. More than 1,500 distinct gen AI applications are in active enterprise use without necessarily the consent of the CEO or of the IT security guard. And more than 3 million DNS queries to gen AI platforms are being processed every single month on one single platform. So, that traffic exists in the logs. It's something that we can check. So, the question is whether anyone is reading it.

So, as we can sum up here is that when we talk about unpreparedness, we tend to mean generally not enough budget or not enough tools or not enough expertise. But the evidence shows that five distinct dimensions, each independent, each exploitable. An organization can have excellent tooling and zero governance.

Perfect visibility and no speed to act on it. Readiness is not a single dial you turn up. It's five dials and most organizations have not found four of them yet.

Last slide, please. So, what can be done or what should we do? So, we need definitely to see for the visibility, we need to know what are the tools being used in the company or the organization. So, approved tools, shadow tools, personal accounts on company devices. So, because we cannot govern what we cannot see. And Cisco found, for example, that 60% of organizations have no visibility into this at all.

So, this is not about replacing the current stack. It's just about adding the layer that sees what your existing tools are blind to. Second thing is about ritual for malware free attacks because nowadays this is something traditional. New attacks are malware free. So, traditional and classical tools are not able to detect these AI accelerated attacks. Rebuilding the detection strategy around how modern attacks actually behave and not how attacks used to behave.

The third is to shrink the response window because we need to reduce the time between when an attack starts and when your organization actually stop it. And finally, we can talk about to train the staff on AI specific threats. That means, for example, one of the reports shows that

only 48% of organizations believe employees understand how attackers use AI. So, that means half of the workforce can't reliably identify an AI generated phishing email or similar.

So, to conclude on a positive note, all the reports at least that I used to prepare this talk show that the organizations that invested in readiness are 70% more likely to detect AI driven attacks. So, the tool is the same. The decision is yours. Thank you.

DANIELLE RUTHERFORD Thank you, Nabil. We have a question from Matthias.

MATTHIAS HUDOBNIK Thanks. Matthias speaking. It's just more a comment than a question. So, thanks a lot, Nabil, for this very interesting talk. I just want to touch upon, has somebody of you heard about OpenClaw? So, this AI assistant every person which is a bit geeky should play around. It's an Open-Source tool. It has like 271K likes on GitHub and you can deploy it fully autonomous and chat with it like via telegram. It opens all the boards. It's a very interesting and very powerful tool. This guy developed it with web coding with Claude Code and now he joined also OpenAI and they're also planning to integrate this.

And it's crazy. You can run it on a Mac mini and it's very interesting. It's like really, yeah, you chat like with a human and it learns based on your content and you can set up it quite easily. So, check it out. It's quite scary and very interesting. Thank you.

DANIELLE RUTHERFORD Just checking there. Do any other SSAC members have any comments? We have a hand from the audience. I'm not seeing any. Andrew Campling, go ahead.

ANDREW CAMPLING Hi, sorry. Thank you. Really interesting presentation. Unsurprisingly, if you go to the RSA conference at the end of this month, AI pretty much dominates the agenda as it did last year, but even more so this year. And to add to the negativity a bit, there was a talk last year how AI is using the DNS to have enhanced DDoS attacks by exploiting weaknesses in the DNS.

So, there's evidence of AI using the DNS as an attack vector. And yeah, organizations are unprepared, don't really understand what the risks are and don't generally have the skill sets. But I guess that's true of DNS as much as it is of cybersecurity. So, we're all in the same space. Thank you.

NABIL BENAMAR Thank you very much. So, the talk was more about how prepared are we because we have a Work Party which is about the use of AI and the relation between AI and DNS Abuse. We have just started this work party and we are preparing the report. And this will investigate and it will show to the audience, to the readership, how really AI is moving things forward, both for attackers and for the defenders. But the thing here is more about how the organizations are facing this, how is the speed, how

the level of preparedness of IT teams, et cetera. So, thank you very much.

BARRY LEIBA Okay. Thank you. Thank you, Nabil. And Andrew, I would say AI pretty much dominates every discussion these days, no matter where you're talking about. Sorry, one more question? Ram.

RAM MOHAN Nabil, so do you think there is something for the SSAC to do here?

NABIL BENAMAR So, yeah, that's something that we are already doing within the work party, the DNS Abuse, and we have started the work, and our goal is maybe to publish one first report, and maybe we can publish more, depending on the acceptance of the first work and because there are really things that are happening and the updates are on a daily basis when it comes to ai and the implementation of ai in everything but I think that we are already there for SSAC and we took really a good thing by creating this work Thanks.

BARRY LEIBA Okay, thanks. Now, Wes Hardaker, one of our newest SSAC members, actually.

WES HARDAKER Brand new SSAC member. Be nice to me. Nobody knows me here.

BARRY LEIBA Nobody's ever heard of Wes before. Anyway, Wes is going to talk about some resolver behavior. Go for it.

WES HARDAKER I'm Wes Hardaker, currently working at Google, not speaking for them. So, I was done some studying in the past couple of months about testing what happens with a resolver when authoritative servers are broken in various ways. And this work came about because of a discussion about .internal, which derived from SAC113. And the proposal was, well, what happens if you just change a name server for .internal to dot? So, you actually tell, you know, the resolver, hey, go back to the root again. So, you sort of create a loop. We'll get into that.

So, I'm going to go over the background a little bit about what happens with names that don't exist because .internal is actually a TLD that doesn't exist, technically. I'll talk about my experimental testing and the results from that and then draw some conclusions.

So, as everybody here knows, the DNS can fail in spectacular ways. A lot of the time it fails because operators fail to deploy their zones properly. There's broken infrastructure. Names that exist don't exist globally. And we're going to come back to that because names can exist regionally but not globally, like within a corporation or without of a corporation. And then internal namespaces have pretty much always existed, right?

So, there are three famous ones, core, poem, and mail, of which SSAC is very familiar with. And then SAC 113 was written to say, you know, maybe everybody should use the same one rather than make up their own. So, .internal was the result of that. So, what happens when a name doesn't exist for something like, you know, something .internal? What actually goes on there in resolvers? And so, when you are resolving something like child. parent, it doesn't matter where it is in the tree, right? The parent can be anywhere deep in the tree, but, you know, there are sort of a couple of cases. One, the parent doesn't exist at all. This is true for .internal today.

The parent exists, but the child doesn't. In other words, you're still trying to look something up, but the parent doesn't know how to answer you. Or parents can be broken. So, the name servers can be broken. They could be broken because you're trying to get the name server for a parent, and it turns out that you can't get the glue because it's an internal bailiwick that there's no glue provisioned. Or there's an external name server that actually is broken and doesn't exist or something like that. And then finally, the last one I'm going to talk about today is a name server pointing to dot.

So, as I said, this is not internal specific, but a lot of the rest of this we'll talk about .internal. So, as I mentioned, .internal today is sort of the parent doesn't exist. SAC113 and the ICANN Board have said we're not going to DELEG it. We won't get into the politics of that today, but this may end up affecting future SSAC guidance on what to do next.

So, what happens when a validating resolver does query to something in .internal? So, there are two halves of this diagram. The left-hand side is the inside the corporation. You're inside your corporation. You query for mail.internal, and you're going to get back a record because you're actually in an environment where .internal exists. You're in a local private namespace, and it doesn't really matter whether it's signed or not. You're going to get some record, and it's going to have some TTL.

Well, if you query from outside and you've taken your device and moved it somewhere else, with an internal non-existing like it is today, you're going to get an NSEC record back saying it doesn't exist. And the time of length that you remember that can be the NSEC TTL, which can be two days long, and a lot of implementations limit you to one day, and it is signed.

So, you have absolute belief that this parent, .internal doesn't exist. So, then the other cases that we're going to talk about, which is not the case today, is what happens when .internal is there but it's an empty zone. It's empty. There's no actual records inside of it. You're querying for some random name inside of a zone that doesn't exist. Then you're going to get back an NX domain. It doesn't really matter whether it's signed or not, which isn't entirely true.

And the TTL that you're going to use, how long you're going to remember that it doesn't exist, is not based on the TTL, the record, anymore because there isn't one. It's not based on the DNSSEC signature. It's actually based on the empty zone SOA negative cache TTL.

We're not going to dive into the details of that, but just there's a parameter in the SOA record that dictates how long do you remember something that doesn't exist. And then finally, what happens if you send a query to .internal and the name server says, hey, go back to dot? Well, clearly that doesn't work, right? You're actually sort of creating a loop there, and you actually get a serve fail back. So, you actually get an error condition back that is not signed in any way. And the TTL for how long you actually remember that serve fail turns out to be very, very short on the order of one to two seconds depending on the implementation.

So, here's the problem. Validators move. Resolvers move. They're no longer stagnant sitting in the infrastructure. So, what happens if you're inside your corporation, you query for something that says, hey, mail.internal exists, and then you go somewhere else, right? So, in this case, you remember this. You've cached the answer now.

So, when you go outside, you're using your cached answer. And it doesn't matter whether .internal doesn't exist, whether it's an empty zone or whether it's pointing to dot, because you've already memorized it, right? So, you're actually going to use this cached registration. Now, that may not be a usable record, right? You may be trying to query to some mail server that only exists internally, so that's a whole different problem. We're not going to go down that problem. But you believe that you actually have the answer for where to go.

And the length of time that you're going to remember this when you're outside in the rest of the world is the length of the TTL for your inside record, right? So, what happens in the other case? What happens if

actually you're out in the real world, and you have queried in the past for, hey, I'm at mail.internal, I need to get to my mail server, and you get back a response that says it doesn't exist? In the case of .internal today, you're going to get back a signed answer.

That signed answer is going to be essentially the length of the NSEC TTL saying it doesn't exist. You can remember that it doesn't exist for a day. And there's some implementation details. I'm glossing over a little bit of complexity here, but essentially that's the result. So, what happens in that case? You go then back internal into your corporation, and you end up remembering that. You remember that it doesn't exist.

So, you're in your internal corporation, and you can no longer actually get access to mail.internal because you're remembering it for an entire day. So, you go through your entire workday. You get no mail read, which might be a benefit. I don't know. And then eight hours later, you go home, and you're still remembering that it doesn't exist, right? And you start this whole loop, and you never get to read mail inside again.

If, on the other hand, we changed how .internal worked, or any parent, and we said it's an empty zone, then you're going to get an NX domain. And the TTL can be controlled, right? Because we can actually create an empty domain with a different negative answer cache that could be short. It could be long. We could control it, though. For .internal being a name server record of dot, you'd get a serve fail, and now it's actually implementation dependent, and it becomes a short possibly one to two second.

And so, one of the things that came up is, wait a minute, doesn't this create a loop? Won't this actually cause the internet to break if we actually publish a name server record to dot? We can't possibly do that. And I thought to myself, well, we can test that. Let's break the internet. So, I took 5,000 RIPE Atlas nodes. We dedicated two domains under two different TLDs, and these just happened to be domains that I controlled, so I did one under .com, one under .games to see, hey, is there a difference between new gTLDs and classic? Didn't expect one, but I thought, why not test it?

I used one authoritative name server, and this is somewhat of a problem because you get large timeouts because it was in one spot on a planet, and RIPE Atlas nodes are all over the planet, so some timeouts exist because of that. We'll get back to that in a minute. And I recorded all the traffic.

So, I recorded all the traffic to my authoritative name server. I got all the data from the RIPE Atlas nodes, and I looked at all the data for these requests that ended up at B-Root because I was able to coordinate with that team.

So, we tested these resolvers in various scenarios. I queried for a name that existed, classic scientific control case. I queried for a nonexistent domain name, so the zone existed, but the domain name itself, the final label, didn't. I queried for a domain name under a non-existing subdomain with a broken internal bailiwick, a broken external bailiwick, and then an NS record with a target of dot. And it's a little bit complex to read, so we'll go through all of that.

And I did all of these three times. So I controlled the TTLs. I did it once in the beginning. I did it within the cache period to see what happens if they should have had the answer cached, and then I did it again after the cache should have expired just to see what was going on. As I mentioned, I recorded the traffic from three different places, from the Atlas data itself, from my PCAPs I recollected at the authoritative server, and from the traffic received at B-Root for those specific domains.

So, let's go through the data. What did it actually look like? So, this is the RIPE Atlas data, and this is sort of a complex graph, but each of the colors end up different answers. So, you have no error, NX domain, serve, fail, and timeouts. This is a little bit complex, so I'm going to boil it down to just this.

So, these are the six test cases that I talked about. The far left one is for a domain that actually does exist. I actually literally created a domain called exists and queried for it. Naturally, we got back mostly no errors. That makes perfect sense.

The second case was for domains that didn't exist, and there were two different types of this. There was simply the parent existed but the label didn't. That's the left-hand one. And we get back mostly NX domains. And then there was another one for the subdomain didn't exist. So in other words, the entire subdomain didn't exist, so you're searching for a.b.parent, and b didn't exist. And naturally, we get back NX domains for that.

And then the final three are broken internal bailiwick, broken external bailiwick, so name server exists but you don't have glue. Name server

exists for a record that doesn't exist. You can't get to that name server. And then the final one is what happens when the NS record actually points to dot. Can we break the Internet?

Here's sort of the same data in a heat map. You can see that there is sort of answers in various different sections. I'm not going to go too much into detail on this, but it allows us to actually look at the raw numbers. And again, same sort of thing, right? The top one is for names that exist, so you get no error back. The middle one is NX domain. And then the final three are the broken domains. So, it's about what we expected.

So, my conclusions from the RIPE Atlas data are about what we expected, right? When a name exists, you get back no error. When a name doesn't exist but the infrastructure is actually answering you, saying it doesn't exist, you get back an NX domain. When the name servers are bad, you get back a serve fail. This is sort of what we expect. There's a little bit of weirdness. We'll get back to that in a minute.

The interesting thing is that the name servers pointing to dot are no worse than the broken bailiwick examples, right? It's still we're still getting answers from the right data saying I couldn't get your answer. I'm still getting a serve fail. There's a little bit of number variation, but it's within a margin of error type problem. So, it didn't go wild.

So, let's look at the PCAPs. What did the traffic actually look like? Could we cause some spikes for breaking the Internet? So, this is the total traffic over time. So, a couple of things here on how this is broken down. There are three peaks for each of the tests. So, the first one is for the dot

com, and you can see the three spikes. I put pauses in between them all. It took about six hours to run.

So, the first one, the first big spike is for the com query. The second one, even though there's a spike, it should have been cached, right? This is actually within the cache period. People were still querying for it, which shows caching isn't always used. And then the third one in each of those different sections there is the spike showing after the cache should have queried, should have been removed. You know, we still then, of course, got a much higher peak.

Interestingly enough, that second should be out of cache is almost always lower than the other ones, meaning some things seem to believe a cache longer than anticipated. I just said that.

So, let's look at the interesting ones, right? We break that down further. These are the top ten source IPs looking for did anything go crazy, right? Is there one address out there that was just like I'm looping? I'm sending data like crazy. I'm especially with the NS record pointing to a dot, right?

And as you can see, there's not really anything. Everything falls off pretty much immediately. All 5,000. So, there's actually more than 5,000 resolvers because a lot of probes are actually querying for multiple resolvers. So, there's actually more than 5,000 different resolver addresses. And then, so I did this test actually three times. In my final run, I ended up with this graph. And so, these are the top ten IP addresses in the final graph. And you may notice that there is one

address that actually did send more traffic over time. And it's queries per minute.

You can look at the numbers on the left. They're not like extreme queries per minute or anything. But one of those addresses did do something funky. I will note that so that final set of peaks, right, is for the name server pointing to a dot. But all the peaks to the left of it are the broken bailiwicks. And it's also sending additional broken traffic for all those too. In other words, even though I would expect a name server pointing to dot to be like the worst case causing a loop, it turns out that any broken name server causes this particular resolver to be sending more traffic. So, I looked up where is that particular address. It turns out to be a resolver in Russia. No qualms to them. It was just, I looked up what region is it coming from.

One of the other things I did look at is the top ten ASes. And so, not surprisingly, some ASes do exhibit more is because they have a larger set of infrastructure. Cloudflare in particular I think was the top one in the other three. So, some of the larger ISPs did generate more traffic than the others. And that's not surprising because more probes are under their infrastructure. So, we got more data.

All right. So, last, let's look at the data that I collected at B-Root. If you look at the requests received at B-Root. Again, nothing dramatic. You see some change over time. The number of queries per minute for those particular subdomains are not extreme. But we did receive stuff from around the world. If you look at the addresses, the top ten addresses seen at B-Root.

Again, very sharp spikes. That one address didn't send anything to us. So, it was probably talking to some other root server operator, not ours. And nothing terribly surprising there. We didn't see anything that evidenced a lot of traffic coming back to at least B-Root saying, I'm continually querying you for the same thing over and over. Which some addresses on the Internet do anyway.

So, what did we learn from this? So, that's the entire set of, you know, infrastructure tests. What did we learn? So, conclusion number one. All error conditions cause more traffic. You know, if you look at, like, the good on the left, the peaks are very small. If you look at anything where the traffic didn't exist, including just the label didn't exist, it caused more data, which is interesting.

Query number two. Broken authoritative DNS servers are the worst. They cause serve fails. They can't be cached. There's a lot more data here to possibly look at in future work. Conclusion number three. Name servers turned dot are no worse than the other errors. I'm not saying that we should do this, right? But the reality is it actually didn't cause breakage that I could at least exercise with 5,000 probes. Conclusion number did I skip one? I may have miscounted.

Conclusion number five. There's no difference between common games. One of my initial things. I didn't really discuss it going around, but not surprisingly, modern TLDs are all treated pretty much the same these days.

Conclusion number six. An empty zone is probably the best. And so, what I mean by that is that if I was going to pick one of these solutions

for should we do something with .internal that just doesn't make it exist, should we actually stand it up in some way to say it doesn't exist, the .internal one, I think, is my conclusion for probably what's best. Because you get a proper NX domain that you can control the cache length for. It's not serve fail where it's implementation dependent. And it doesn't cache for an entire day like the current deployment.

Conclusion number seven. Strange things always exist. That one IP address out there does do weird things. But as I said, it does do weird things for like a bunch of the tests, not just the weird one.

Conclusion number eight. Stranger things always exist. Why in the broken cases on the right-hand side of this heap map are we getting no errors when the domain doesn't exist, the name servers are broken, and things like that? A no error response in those cases doesn't make any sense. That's future work to sort of dive in and figure out what's going on there.

Conclusion number nine in the last one. Even stranger things are afoot at the circle K. I got 26 questions about A6 records. That's not quad A. That's A6, which is like the oldest DNS record that's been obsolete for over a decade.

I got queries for CNAME. Why is somebody querying not for a record name with an A record or a quad A, but literally querying for a CNAME? I got query for an MX record. I never asked anything to send mail. Why am I getting a query for a mail server? And then I got a query for a text

record, too, which I suppose I could think of. Maybe it was the mail servers trying to figure out if I had SPF records. I don't know.

So, with that, any questions except for the last three? You don't get to question about the weird stuff because I can't answer that.

BARRY LEIBA And so, we're not going to take questions. We do have a couple people with questions, but we are out of time. So, now I know what Wes is going to be doing in the upcoming coffee break. So, please buttonhole Wes during the coffee break if you were one of the people who had questions. And we're back here in this room in half an hour for another session. Thanks, everybody.

KATHY SCHNITT Scott, please stop the recording.

[END OF TRANSCRIPTION]