ICANN86 Seville | PF – ccNSO: Tech Day (2 of 2)
Monday, June 08, 2026 – 11:45 to 13:15 CEST

EBERHARD LISSE: Welcome back, everybody.

CLAUDIA RUIZ: Hello, and welcome to the ccNSO Tech Day session. My name is Claudia Ruiz, and I, along with my colleague Joke Braeken, am the participation manager for this session. Please note 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.

Please observe the following guidelines to participate in this session. They will be posted in the chat for your reference. During this session, questions or comments submitted in chat will be read aloud if put in the proper form as noted in the chat. Interpretation for this session will include Spanish and French. If you would like to speak during this session, please raise your hand on Zoom.

When called upon, virtual participants will be given permission to unmute. On-site participants will use a physical microphone to speak. Please state your name for the record and the language you will speak if speaking a language other than English. And please speak at a reasonable pace to allow for accurate interpretation.

Thank you very much. And with this, I will now hand the floor over to Eberhard. Thank you.

JAROMIR TALIR

Okay, I'll take over. This is Jaromir from CZ.NIC. I will chair the second part of this Tech Day. We have five speakers, five presentations. They are all here in person, so I hope we will avoid the issues with remote. And I hope we will have more luck with the power. But without further ado, I will hand over to Steve Crocker to talk to us about the recent deep evolution with his project, Jake. Steve, the floor is yours.

STEVE CROCKER

Thank you. At the last meeting, I presented work that we've been doing. and in cooperation with TWNIC on developing a framework, a fairly broad and comprehensive framework, to implement multiple policies related to WHOIS access to registration data and so forth. So this is going to be an extremely short update on the progress of that. There'll be a slide or two describing what we're trying to do. And then hopefully, despite the fact that there are technical problems in Tech Day here, we'll see if the demo actually will work, and then we'll move on.

Project Jake is the name of our project. We're doing this pilot with TWNIC, as I said. I hope to be able to give you a live demo with test data at this point, not actual data from their system, but nonetheless, using their actual system. So it's real from that point

of view. We have the basic software working. There's always lots more to do, packaging, user interface, and removing extra commas.

The process involves both technical aspects of presenting requests and processing them, and non-technical aspects of making agreements so that all the parties know what to expect and are bound by various conditions. And all of that is put together in a way that hopefully is efficient and smooth and clear so that expectations are understood. So the key elements that you have, in broad terms, are people who have the data and people who want the data. We call them data holders and requesters.

The data holders have to make, in principle, a complex decision about who's making the request, what it is for, and whether they can trust that the requester will behave properly. And even beyond that, what happens if something goes awry, either accidentally or on purpose? So agreements are a part of the system. And the more you can put into those agreements ahead of time, the more you can reduce the amount of work and the complexity of actually processing a specific request.

One element of achieving that level of efficiency is for groups of requesters and groups of data holders to aggregate themselves to the extent that they have common interests, common fate, and so forth, into requester groups on the one hand and data holder groups on the other hand. And for the agreements to be between those groups.

There's no architectural control over how many groups there are or what their boundaries are. So if one thinks about law enforcement or intellectual property, attorneys or whatever, there's nothing that says there's only one of each of those kinds. And there's nothing that says that once established, they have to be maintained like that. They can grow, they can disappear, they can merge, et cetera, et cetera. The agreements have to have the common-sense part of it. What is the purpose? What are the permitted uses? How do you control and protect the information? And then what data elements are expected?

And there's quite a bit of additional structure to it, including a slightly expanded notion of the sensitivity of the data, not just public versus private, but for having listened to an awful lot of discussions about those kinds of distinctions, there's an argument to be made that there's actually multiple levels of privacy, not a huge number, but more than two in the sense of its not just public versus private. From an architectural point of view, we're talking about software on the left, which is in service of requesters, and software on the right that's in service of the data holders.

Data holder is a term that encompasses registrars, registries, and resellers, if you want. The common factor, of course, is that that's where the data is, and this is adaptable to both thin registries and thick registries. And on the bottom of each side are the operational pieces of software. So a requester uses a requester service to

initiate a request, and it goes directly over to the data holder, not through any intermediaries.

And there's an OpenID server involved that has the credentials, and there's a bearer token that's included, and the data holder can check all of that. All that happens at internet speeds and is efficient. Above them are the group management functions, creating groups and creating agreements between the groups, and adding or removing members. So that's the basic simple structure.

I want to hopefully, if things work, I'm going to give you a demo of making some very basic kinds of requests between a requester and a data holder. So I have in mind three quick demos. One is, if somebody is a member of a requester group, and so there are already agreements in place and an identity and authentication structure, two kinds of requests: one for public data and the other for non-public data. And I'll show you the distinction.

And then, assuming all that holds together, what happens if somebody is not already a member of a designated user group and wants to make a request? What process do they go through? And I'll give you a taste of that. So there are two windows here. The one on the left is the requester module, and the one on the right is a piece of the data holder module. And we'll just make a simple request. The dimension that we're using to test data at the moment.

So the domain name that we're looking up is legal-normal.test.tw. We're a member of a group, and that group has access to a series of

possible kinds of requests. And here's what happens if we make the lowest-level request, which would just give us some public information. There we go. And what comes out here, you know, the registrant, billing, admin, and tech contacts, and only the email information for the registrant and the billing contact, and only the email and the telephone numbers are included in this particular kind of request.

And on the other hand, if we make the same kind of request, but with higher-level access that we're permitted to make -- so this gives us everything -- then we make the same request. You see that all of the data is filled in here. And those are pre-authorized kinds of requests. And so it goes just as smoothly as that.

Now, there's an awful lot of situations where the data holder is saying, "Don't want to automate that, want to look at things, want to see before I make a decision." So I'll give you a couple of versions of that. One is the same sort of request, but looking at what's going on. I'll make this request, and it says, "Please fill in the required fields." In this case, all that's being asked for is a case number. So we'll make up a case number.

I will just make up an arbitrary thing. I'll call it Paris nights. Why not? And we'll search for this. We're still working on the user interface. This message is no longer relevant; we'll make it go away. And now over on the data holder side, which is on the other side of the network, I'm looking at the pending request panel.

So imagine that you're in the offices of the registrar or the registry, and here's a request. And what comes up is a panel that gives you the information about the request. And here is the case number that I came across. There's Paris nights there. And if I approve that, then in relatively short order, it gets processed on the data holder side, and then comes over here to the requester side, and it got approved, and all the data got filled in again. So that's a very small example.

Let me give you a more complicated example. Let's imagine that instead of being a member of a designated user group, this person is presenting himself as unaffiliated and says, "I want to get some data." And we'll go right to a higher level of access. And so let's say that he tries to make that, we'll make the same kind of request. And it says, "Oh, please fill in all these required fields." What fields are there? Well, what's the name of the requester? The name of the requester will be, we'll just say, Elmo Everyman. And what's his email address? I'll just say ee@ff.gg. It's obviously not real. And now we've got to start sending some other heavy-duty information.

So Case 714 is the file, and I've got to fill in some other things. I've got to choose a description of good faith, and I've created these in advance because I know that I'm going to be asked for these sorts of things if there's decent documentation. So the good faith thing I've called Legitimate declaration, and I'll open that. And now I've also got a compliance statement.

And the compliance statement says, "Can I actually protect the data?" Yes. And there's my documentation that can do that. An accuracy declaration. And for demonstration purposes, instead of a file, this simply asks for a string. And I'll type in my declaration that this is accurate, because I'd said so. Authentication failed. I've got to sign in again. This is a real demo.

EBERHARD LISSE We're a little bit over time, Steve.

STEVE CROCKER Yes, if you say I have to quit, I will. In the interest of time, I'll simply tell you what's going on. The files are transmitted, and then there's a panel that becomes available, like you saw before, and it includes access to those documents, which you can download or you can look at and make a decision, yay or nay, and close up the thing. I'm conscious of the time I'm consuming, so I will simply abort this. And get back to the tail end of the slides that I wanted to show you, which includes how to find out more information. So those are the demos.

We have a room set up in the Congresso, the hotel next door. And we're doing demos this afternoon and tomorrow, and on other days by arrangement. Contact me. I'm happy to get back to you. Relatively short, but opportunity to probe into any of this that you want. And we're continuing. If you want to get engaged, there are multiple ways to be involved. If you are a requester or a

representative of a group of requesters. Or if you're a data holder or comparably want to represent data holders, or any other role that is involved and interested in all of this.

We're continuing to build and expand this, and as I mentioned at the beginning, we're in trials now with TWNIC. And as that progresses and the pieces of this get smoother, we'll be happy to work with anybody else who's interested. Thank you.

EBERHARD LISSE

Thank you, Steve. I have a passion for live demos. And congratulations, it worked. We don't have time to follow up on questions, but there is a link to the email so you can contact Steve directly. So we'll go to the next presenter, who is Luis Diego Espinoza. He will talk about Proxmox. Please, Diego, the floor is yours.

LUIS DIEGO ESPINOZA

Good morning, good afternoon. My name is Luis Diego Espinoza. I work as an advisor for the Namibian registry. Today, I want to share a technology review. This is not a report, but an evaluation of Proxmox as a practical open-source virtualization platform. Let me set the context. A small ccTLD has a specific set of services. It must be authoritative DNS, WHOIS, RDAB, portal registry, etc. And some monitoring. The challenge is doing this with a small team with a limited hardware budget, typically two or three physical servers and two people, maybe.

Proprietary hypervisors like VMware or Hyper-V, they used to come with a per-socket license or per-VM license costs, and sometimes hard to justify for an organization of a small size. And cloud-only approaches introduce the dependency and recurring costs that can be unpredictable. The idea here is to use, for example, Anycast for DNS, a managed service offered by several providers for the most critical public-facing services, while Proxmox can be used for hardware and everything else -- the backend, the portal, database monitoring, etc. Let's start with the architecture.

You can see on the left what the Proxmox layer cake is. At the bottom is the standard Debian Linux, a very well-known operating system. The bottom will be the storage, ruled by ZFS. Then, the Linux bridges for VLANs. Then the next layer is Debian. And then patched the Debian kernel by Proxmox. Then, for this, you will have the KVM and QEMU for built-in machines and LXC containers. Then you will have Proxmox web user interface and REST API. What are the features, benefits, and security? Why Proxmox? On the operational side, the most important community repository is fully functional.

There's no future gap between the paid subscription and the free version. The subscription just gives you access to enterprise updates and commercial support. That can work for many well-supported companies. You can run both virtual machines and lightweight LXC containers on the same host, managed by the same interface.

And here's an interesting feature about Proxmox versus VMware, for example. In VMware, to manage several nodes, you need vCenter, a different software. In Proxmox, you will have all on the same web interface. On the security side, KVM hardware installation is specific. When VM runs, the CPU utilizations like Intel BT or AMD B enforce hard boundaries between virtual machines and the hardware level. This is very important because if some machines escape vulnerability, they will not affect the other virtual machines, like in Docker containers, for example.

So on top of that, Proxmox has a per-VM stateful firewall building, two-factor authentication on the management interface, and a full audit log and everything, every action with the user with a timestamp. This is what a practical service layout looks like for a small CCTV, for example. On two physical layers, you will have proxmox01, one physical layer, and proxmox02, the second physical node.

And inside each one, you will have several virtual machines. For example, hidden DNS, PostgreSQL database, and monitoring software. And you could have RDAP, WHOIS, a web portal mail system. You will have a virtual machine for every service. On the other side, the DNS will be provided by Anycast DNS or a very well-known infrastructure that can support those queries on the internet. Then the hidden master can live inside the Proxmox and can share the transfer with the AXFR, and the Anycast cloud will be

providing support. Ceph RBD is a virtual storage that is shared across all nodes.

If you lose a node, for example, the VM keeps running because Ceph by itself is very resilient and protected. The PBS system is an integrated backup for Proxmox. You can schedule the duplicate off-site data backups. And you can easily scale out later, adding third, fourth, or fifth nodes. So you can have a full HA availability with fencing. Let's talk a little bit about fault tolerance. Snapshots are one of the most operationally useful features in Proxmox.

Before any upgrade or database migration, update in one of the nodes, one of the BMs, for example, or a configuration change, you can take a snapshot. This is a very well-known service on other virtual platforms, also. This captures the exact state of the VM disk and optionally its memory, and if the great frame breaks, you can rollback easily with one command. It takes around 10 seconds to get the snapshot of the VM and again, another 10 seconds to restore it in case of something failing.

Cloning is another feature. You can create a test environment or development environment for a production VM in seconds or minutes, maybe. You can test the configuration change, verify it works, and you can apply the same change on production with confidence. For backup and disaster recovery, Proxmox Backup Server (PBS) is the tool that takes incremental deduplicate backups. Only change blocks are sent to each run, which makes it efficient even over the one link.

You can set rotation policies, keep daily, weekly, and monthly backups automatically, run on schedule, and automatically you can create individual files from inside IBM Backup without restoring the entire disk. This is very practical. And you can use PBS for off-site backup, also. About the storage backends, we can have a local directory in Proxmox that we may hold the ISO image for install, or container templates, or PBS backups. Then you can have a local LVM, it's a thin-provisioned single-node host, and then you will have the ZFS.

That is the shared file system where you can take instant snapshots. And you can use ZFS Send for replication to a peer, and Ceph RDB is the one you see on the right. Sorry if the image is too small, but it's a pool. It is shared across all the nodes. Every VM will have its own hard disk, for example, in this pool, that will be protected across all the nodes.

And that will enable easy live migration of virtual machines between nodes because the storage is centralized and protected without using NFS, for example, that sometimes is not very efficient to manage. Proxmox backup server has incremental and duplicate. It only sends change blocks in each run; it does not send the whole copy of the backup. That saves a lot of space, and it's very efficient. You also have retention policies, as I mentioned before, and file-level restore.

On the clustering side and live migration, moving one VM to another node will be done on the command line. And this is an

example of the command line you will use to do this. I like it a lot when I can do things by command line because it's very easy to just log into the system and run, or very easy to script. This is one of Proxmox's most liked features in my point of view, because everything you can do on the web interface, you can do it on the command line. And that is not the same for other platforms. So it's very useful.

You can do things on the command line. Corosync cluster allows you to auto-restart on an old failure. HA groups set which VMs will always be running. Fencing is just IPMI, and iDRAC is the platform for hardware management. On the other side, again, DNS will be covered by Anycast.

Scripting and automation are two of my favorites. You can use Pvesh, it's like VMD's ESX CLI on VMware. And you can do the same, Pvesh, as in the underline for the REST API. So in the example, you can see Pvesh get the name of a node. And that will be output in JSON format, the configuration of the node. And you can edit and modify easily, so you can keep that as a backup.

And for example, with Pvesh create, you can create a node easily, or you can get information, stop, start, or do anything you need to do with this command line. With cloud-init, you can create templates for setting up BMs. This is a good feature for ISPs or for data centers that should provide automation in the creation of virtual machines. So you can preset the SSH key, for example, for a

customer, or you can provision many VMs at the same time with just one command using cloud-init.

Also, you can use Terraform. I don't have too much experience with Terraform, but you can use it, and you can use Ansible to automate this. But at the end, I like the scripting thing, and you can automate anything using scripts. This is a complementary thing that could be useful for Proxmox. From the point of view of inventory, that could be useful for many organizations that have a clear inventory of what they already have. For the inventory, we use this one. I have been using GLPI. It's a French software, open source software. And very useful to keep track of not only hardware, it's useful to keep track of software.

So I found that the GLPI agent running on Proxmox, because Proxmox is Debian at the bottom, can extract all the information about the VMs installed on that Proxmox. Very useful information, and you can keep track of everything and create reports not only of the hardware but also of all the virtual machines installed there. And it's very easy to implement.

I mentioned this; maybe it doesn't look like a big thing, but I tried this with VMware, and it's not possible to install a GLPI agent on VMware, for example. You need to figure out another way to extract a report of inventory on that system. For monitoring, I was playing around a little bit with Grafana. It used Prometheus and also InfluxDB to pull information and present some dashboard. And again, because this is Debian underlying, you can run exporters and

create a dashboard. And when I checked Grafana, there are a lot of real existing dashboards for Proxmox, so you can have a live view of what is going on.

This is an example of one of the dashboards that already exists. You don't need to set up that dashboard. It's already there. You just download the dashboard and connect the exporter of Proxmox, running the exporter on Proxmox, and you will have a live view dashboard of what is going on with the system.

Obviously, in Grafana, you can also create alerts and reports, and these alerts and reports will be reported to Telegram, for example, or SMS. And then you can keep an eye on the structure using this open source software. So some notes about the sterilizing costs and enterprise features, fully open source, that is Proxmox, Debian LTS base. Most of you will be familiar with the kernel, with the command line, and with the operating system by itself. It has long-term support, it's secure.

On the other hand, Anycast DNS offloads, so you will not try to run the DNS resolver by yourself. Probably it is a better idea to just hire Anycast. REST API, you can use bash, Terraform, and Ansible, and you can automate everything. That's a great thing when you have a small budget, a few people, and the best thing is to automate everything. This is a very nice feature: the Ceph storage system and the PBS backup system. You can share storage across all the nodes. If you have four or five nodes, every one of them with its own storage, you can create this virtual storage across all nodes. And

the Ceph itself is so resilient and very well backed up. It's protected. So that's very nice.

And with PBS, you can create remote off-site backups, for example. It's a very good thing. On the side of monitoring, you will have something like GLPI, Grafana, and even Monit. Monit is a very small program. It is a watchdog that can check system services and report things. So that's the presentation I have for. Thank you very much. If you have any questions?

EBERHARD LISSE: Unfortunately, we don't have time for a question, so we will again have to leave it for direct communication with Diego. We will go to the next presentation, which is from Kristian from .se, about the unified way to do registry lock.

KRISTIAN ØRMEN: Thank you very much. So later in the session, I will have a couple of questions on Mentimeter. So if you want to, you can make yourself prepared to answer those. That will be in the end. Also, since we don't have much time, I just want to say that I will be waiting here after the session. So if you have thoughts, questions, and so on, just come up here also afterwards.

CENTR is the Council of European National Top-Level Domain Registries. We have a lot of different working groups, meetings, and so on. But we also created this concept called a task force, where we take a specific topic and gather people from different

working groups, both technical, business people, and legal people, together on a specific topic.

We had a meeting where many were frustrated with registry lock today, either because they didn't sell enough, or registrars wouldn't sell it, or they were not secure enough. There were many different solutions and also problems. So these are the members of the task force. It was 10 different TLDs that participated with input and work in the task force.

This doesn't mean that all these 10 TLDs will implement what I'm showing today, but all of these TLDs have been included, and some of us have already decided to implement them. When we started the task force, we, of course, set some goals for what we wanted to achieve. Here's just some of them, and I'm going to go into more details in the next couple of slides.

But one of the important things was that it should be possible for a registrar to sell registry locks across many TLDs with the same marketing material, same description, and so on. We also wanted to make it possible for registries to lower the cost, since many registries have a very high cost for registry lock. And we think that is one of the reasons why they haven't been selling enough registry locks.

Also, we wanted to standardize it since registry locks have so many different phases, and that also makes it difficult. Many registry locks have very manual implementations and processes, and only work during business hours. We wanted to see if we could

automate it and still keep it secure. So, how we worked with this is that we started collecting all the different registries, like how do you do registry lock today? And we were trying to find the most important features of all the different designs. Then we also contacted the important registrars, both some that were selling registry locks today, but also some that were not, to find out what they need if they should begin selling registry locks.

We created different personas, both registrar and registries and registrants, to try to catch all these different cases. We also did have some interviews with some registrants who use registry locks a lot. So based on that, we could define the business requirements in more detail and started to draft specifications, even like EPP extensions down to the details. And where we are now is that we have published a registry lock document. It is available in the CENTR library on centr.org.

I, just last week, made sure that it is now even made public. So that's why I didn't have the link in the presentation at the start. But CENTR without the E and then .org. And library search for Registry Lock, and you will find the full documentation. Or else, you can come here afterward, and I will definitely help you find it.

And on the link on the slide, you can find the draft of the EPP extension that was also created as part of this work. I can start talking about the next slide. So, registry locks really have a lot of different designs. In .se, .nu registry, where I'm coming from, registrars can deactivate registry lock in the registrar web pages.

So personally, I think it's a bit of a stretch to call it a real registry lock. Usually, in most other registry lock designs, the registry will have to deactivate somehow. If you look at other registry locks, they have paper forms. They call out. Some of them use EIDs. It's very complicated, and many different procedures. But of course, we all have this goal of securing domains and securing against unwanted changes.

If you look from a registrar perspective, if you are a brand registrar, those are usually the ones that sell registry locks. If they have a customer with 20 TLDs and the customer comes in and says, "I want registry lock on all of these," then there's a good chance that they would have to explain 20 different procedures on how registry lock will work. Some of them will be very cheap. Some of them will cost maybe even thousands of dollars. Some of them will require verification of the actual contacts, down to the passport copies, and some of them are just like EPP, and it's activated.

So what we really want to achieve here is that the registrar can go in and say, "Yeah, for these 20 TLDs, this is how registry lock works." The pricing may be a bit different, but the processes will be either exactly the same or at least almost. So we also had this question: Can we automate it and still keep it secure? And that's really the tricky part because registry lock, we don't want any unvetted changes. We want verification that the requests are okay, which can be a bit more difficult in an automated way. But we think that we succeeded in having a quite good design where we still keep it secure, but we automate most of the processes. So on the next

couple of slides, I will try to go to some of the most important features.

So, in order, in our design, to activate a registry lock, the registrar will activate it via EPP. The request will then be pending, basically 1001. The registry will contact the registry lock contacts and ask for approval to activate the registry lock. Most registries are open in the draft on how they will do it, but most will most likely send an email to these registry lock contacts, ask them to approve, and ask them to activate 2FA. And when all the registrants and the lock contacts have approved, the domain will be locked. So then in the future, after a domain is locked, if you want to change a name server, for example, just as an example, in many registry locks, designs today, you will unlock the domain and then do the changes.

In this new design, you do it the normal way, so that the registrar will send the update command to the registry via EPP. Again, just like when we activate it, we are going to set that request to pending, and when we have the request pending, we will contact the lock contacts. Again, most registries will do this by email, and they will ask for approval for the change. So they can either of course not approve it, or they can approve it, or it could time out. And the timeout is also a policy that could be set per domain. When it's either approved or not approved, the system will notify the registrar via poll message, and if it's approved, the change will go through.

So, just to explain, like in the old model, for most registry lock designs, you would unlock a domain. When you unlock a domain, you can make all changes. You could delete a domain. You could change name servers. You can owner-change it. So it's completely unsafe when you unlock it.

In this new design, the registry lock contacts, when they're asked to approve a change, will also be notified of what change they are approving. So in that sense, it's a lot more secure since they know what they are approving instead of just completely unlocking the domain and having it vulnerable to whatever change that could happen. So we see this as a much more secure approach to registry lock, and the contact will be aware of what's happening during the whole process.

I talked a lot about registry lock contacts, and one of the reasons we wanted to have registry lock contacts is that many registry lock designs today have these contacts in different ways. So we are then saying, just to make sure not to have issues, we always want at least two registry lock contacts. So if one is missing, the other one can approve. But we could also think that some domains may need a bit more security.

And we thought a bit about how it works with net banking. And if you have a company account, sometimes you say, for example, that at least two people need to approve. And we said, this, we could implement as part of this design as well. So on each domain, you can set a minimum quorum. It could, for example, be one. And

then you would need at least two contacts, or you could have a quorum of a minimum of three, and then you would need at least four contacts.

But you could also have like six contacts, but only needing a quorum of three. And in that sense, you make it less problematic when people are on vacation and so on. If we go back to the approval process for changes, we decided that if only one contact denies a change, it will immediately fail. But for approval, you need the full quorum.

One of the things when we talk to registrars and also when we talk to registries is that even though this is maybe not the thought of many registry lock designs, it was noted that many registrars will be the registry lock contact on behalf of the registrants. That's what's happening today. So in the design, we are kind of trying to think, if we know it's going to happen, can we make sure that it would also work?

And in this case, we think it works because you could choose to have the registrar as your lock contact, and in that sense, we allow many ways registry locks are being used today, and the registrar is fully informed and has chosen this themselves. So one thing, in the policy, it may allow for the registry. We haven't put it in strictly.

But at .se we have a DNS scanner, and we allow automatic updates. For example, DNSSEC records. So this is a policy question. We haven't decided exactly for .se yet, but I think we are going to allow DNSSEC updates via our DNS scanner. We think that's sensible,

automating even with registry lock. So, as I said earlier, it doesn't necessarily mean that all these 10 registries have decided to implement this new design.

When we were done with the task force and doing the design, we started talking about who would implement it. Some, of course, have long development roadmaps and can't commit to doing it now. But some are committing already. And I can say from our side that .se and .nu will implement this design. We have it in our roadmap, and we will start development next year.

I also know at least two other TLDs that have said that they are like 99% sure that they are going to implement this design as well. And of course, one of the main reasons I'm sitting here today and talking about it is because we would love to have more people on board. We would love to have more registries implementing this same design. And I really would love for you to come and talk with me after if you need more info or if you need to sit down and have a meeting and go through the more nitty-gritty details later as well.

So, yes, let's jump to the Menti, and we can go back and see if we have time for questions. We don't have time for questions, so that's good that we jumped at that. So the first question here, and I know that some people, and maybe we have more than one in the room, but just to see how many registries provide registry locks today, I'm just going to give it a short time before we jump. I see that 46 people logged in, more than 50.

So let's get the answer a bit more up before we continue. Well, I think it's clear to say that many registries provide registry locks today, but also, there are a couple of No's, so definitely not all. And even some that have it in the roadmap in the future. So, for you who have it in the roadmap, I really, really hope that you would look at this. So, for those of you who do not provide registry locks, so everyone who said yes before, please do not answer this one. If you do not, do you think you would adopt this proposal as I just presented? One Yes already. That's fantastic. Two? Amazing. Three, even better. Four? Seven? Holy. So this is really amazing. I hope to talk with a lot of you.

Also, those who don't want, I would love to understand why you don't want. I know from some that they said it's a huge change in our registry system because of the asynchronous changes. And I understand that. But let's talk more about it. So, for those of you who do have registry lock today, would you consider changing your registry lock to implement this proposal? And basically, only the yellow is negative, so all the others are quite good answers. For those of you who do currently provide, which lock do you think you will consider adapting this proposal to? And a lot of people want to think about it, some want to do it, some don't know.

And also, a couple of Nos, and that's understandable. But again, let's continue this talk. We don't have time for questions now, but I will stay here when the session ends, and please come up and ask questions. And also, you can find my email in the agenda. You can

find the whole documentation on the CENTR website in the library, just search for Registry Lock, it's the top one. Thank you.

EBERHARD LISSE Okay, thank you, Kristian. The next speaker is Jens Hoffrichter. He will talk about DNSSEC at scale.

JENS HOFFRICHTER Yes, hello, my name is Jens Hoffrichter. Today, I am going to talk about DNSSEC at scale, enabling the signing of 5,500 zones in the real world. This is a project we have been doing and are still doing for a large customer of ours, where we are trying to enable DNSSEC signing across the whole portfolio. I held this talk at DNS-OARC a couple of weeks ago. So please bear with me if you have already heard it. And this is a talk about the perspective of a DNS operator, so most likely one of your users or customers. So just a couple of words about me.

My name is Jens Hoffrichter. I'm the managing director of p-square. We are a consultancy specializing in DNS, SMTP, and DDI infrastructure. And we work mostly for enterprise customers in highly-regulated environments like automotive, banking, and manufacturing. About this project, this is for a major automotive company with a large internet DNS portfolio. They have about 5,500 DNS zones across a highly diverse set of TLDs. We had already migrated the DNS provider a couple of years ago, and we had

DNSSEC on exactly one of these domains when we started with the project at all.

The mission here was to enable DNSSEC signing across the entire DNS portfolio of this customer. But for us as DNS operators, DNSSEC is often a no-brainer because it just increases the security of your DNS. But even getting the go-ahead for this project required significant internal convincing of the customer. It just wasn't seen as a priority. I more than once heard the argument, "The rest of our security stack is already that good, we don't need DNSSEC." What people often don't understand is that even if the rest of your DNSSEC is very good, if people don't get to it, it's not very useful at that moment.

The green light for this project finally came through a BSI requirement. BSI is the German Institute for Cyber Security, especially for mail security. Every mail domain in Germany is supposed to have DANE in the near future, and DANE requires DNSSEC. This was finally what made the customer give us the go-ahead for this project, but also with a slight caveat that we do what doesn't cost money first. For us, as I said, for DNS operators, this is a no-brainer. It is not a no-brainer for enterprise decision makers. So the theory for enabling DNSSEC is simple. You enable DNSSEC signing. For us, it is just basically setting a checkmark on the zone at our cloud DNS provider, get the DS record, submit it to your registrar, verify, and you're all good. But now, do that 5,500 times.

The first surprise we had been getting was that we could, for the large registrar we were using for managing our domains, for a large number of zones, we could not submit the DS records there via API because it really depends on the registry if you can do the DS submission there via API. And especially smaller ccTLDs apparently do not support it. And the policy for our registrar was that if we need to do it manually, we are not doing it. If we cannot do it via API, we cannot do it.

So, for roughly 200 domains of this, we didn't have any automated path for DNSSEC enablement with our existing registrar, so we needed to transfer those to another one, which actually supported manual submission of DS records to the registry. For some ccTLDs, even a domain service provider, I'm using this instead of a registrar here, but a lot of ccTLDs are very picky with which registrars they are working together.

I'm taking China as an example. Our domain service provider didn't have a direct business relationship with the Chinese registry, so we needed to go through an intermediary, which was, in this case, Brandma. And there definitely API submission of DS records didn't work, so we had to do DNSSEC enablement via email. So emails being forwarded, DS records being copied and pasted, and this just led to misunderstandings and errors, and these errors really did happen in the real world. Especially when you're working there with intermediaries and forwarded emails, you cannot be sure

when DNSSEC is being enabled because people work in different time zones.

We had a problem there in Brazil. We made a mistake when submitting the DS record to the registrar and then to the registry. The mistake was detected by Brazil the next day. But by then, we were not working anymore, so I could not correct it until the next day. And this led to an extended downtime before we could even get the domain working again.

So we learned that we needed to schedule the NSSEC enablement around the business hours of all the parties involved in this process. And especially if these are really important, and email domains, this is a really critical step. And if it is really a big change, we learned to do it together in a live call -- in the Zoom call, in the Teams call -- where we could do the enablement of DNSSEC together with the registrar, so that we could verify the success immediately.

This is something which, if you think about it, is very clear, but it is not entirely visible at first moment. Because even if you run your DNS with very low TTLs -- I mean, an hour by an hour is normal. Ten minutes or even five minutes is getting more and more common here, but you don't control the TTLs of the DS records in the parent zone.

So any mistake you make here could take a long time to roll back. And we have seen a lot of TLDs which have a day of TTL on their DS records. And if a mistake happens and you need to roll back, this

could take quite a while before it propagates through the internet, through all the caches. And if you take that, then together with the time zone issue I talked about before, that can mean days of impact if you make a mistake here.

Our main registrar basically takes the DNSKEY as the data for a DNSSEC enablement. So we honestly didn't think much further about it and started sending out the DNSKEY to the other registrar, and therefore through email chains to enable DNSSEC there. But of course, the registry needs the DS record, which is a hash of the zone and the key. The DNSKEY might be the same for every zone, which it is for us because our DNS provider signs with the same DNSKEY for everything.

But here is another manual step we didn't have figured out before, where the computation of DS records needs to happen manually by someone who is maybe not doing it that often, and this can lead to more errors. So for us, it was really that we don't assume what works as one registrar works at another.

Every registrar there is different in what works for them. This one hurt a lot. The registrar portal for the .cloud domain had an ambiguous field label. What we noticed then in our root cause analysis of this outage, it read something. I haven't seen the portal myself. It was just reported to me by our contact at the registrar, which contained a DNSSEC key, something like that, instead of the DS record.

So the registrar put in the DNSKEY, which we had sent him, instead of the DS record. And that broke the whole .cloud domain for that car company, which was especially painful because all AWS production domains were delegated under this special domain. So we had an outage there on big production, which was very painful for us.

And later, what I found a little bit painful to see as well, was that this contact at the registrar had calculated DS records manually and was really asking me if the DNSKEY was the same, why they were different for the different zones, because he simply didn't have the understanding that the DS record contains the hash of the zone name within the DS record.

The enablement of DNSSEC counts as a domain update at most registries, and especially for us, it was this that we had registered domains years, maybe decades, before. They hadn't really gotten updates. The name server stayed the same. The contacts stayed the same.

But DNSSEC enablement can trigger the registries and registrars to wake up and enforce that your domain registration is updated towards the current standard, which might mean updated admin contacts, updated trade registry numbers, and other requirements which might have changed since the initial registration. And this really blocked the DNSSEC rollout for us that we couldn't predict in the beginning, and which wasn't quite clear when we started this project.

This was kind of clear to us in the beginning. When I talked about this before, this is what a lot of people found surprising: that not every update on every registry is free. Sometimes updates cost money, €30-€50, I've seen for an update. As I said before, the DNSSEC enablement counts as an update. And especially if this is under TLD, where we had a lot of domains we were managing, this can easily multiply the cost for enabling DNSSEC. If we have 100 or 200 domains under one TLD and every update costs €20, I can think how much it is and which then needs internal budget approval at the customer, which is not a technical discussion you need to have.

Additionally, DNSSEC introduces new resource records. So if your DNS is licensed by a resource record, which it is for us, this might put you to limits on other ends which you hadn't expected in the beginning. And we also come across a couple of TLDs where we simply can't do DNSSEC. One of them, the .ae domain, the registry simply does not offer DNSSEC at all.

But on a couple of others, I've just picked out the Korean registry here; they only accept DS records, which have been created by algorithm 8, RSA. They don't accept DNSKEYs, which have been created with algorithm 13. Our cloud DNS provider signs exclusively with algorithm 13, so we need to wait for one of them to change so that we can even enable DNSSEC for this one. And I doubt that our DNS provider will go back from 13 to something more insecure.

This was the strategy we finally settled on to get DNSSEC on the highest number of domains. We started with the mail domains. Those gave the biggest impact for us. There was a very strong compliance argument, even if it cost money to do this, because there are federal requirements we need to fulfill to get DNSSEC on. We grouped it by registrar and TLD to batch things to minimize the overhead.

Of course, we did what we could with the API. The API domains were very easy to enable, where we didn't have to do any manual overhead. We scheduled around time zones. If an update cost money, we just postponed those updates and didn't do them right now, so that the cost discussion didn't block the whole project.

Currently, we are about a year into this project, I have to admit, and we have now about 2,500 domains of the 5,500 signed, which is not that much a problem of what we have here right now. But we are using a proprietary feature of our DNS provider where a large number of domains are linked to a single domain, which costs us just a couple of resource records in our allowance.

Unfortunately, DNSSEC signing for these linked zones is not supported. So we need to put them all into their own configuration. These are zones which have like five or six resource records in there. So they are really just parked domains. But those are a lot of zones that would multiply our resource record usage. It would increase it by 5,000 or 8,000, last time we checked, and we simply don't have the allowance to do that right now.

The key takeaway for us, DNSSEC is no longer a technical problem. We have solved the technical side of this equation. Right now, it is a registrar logistics problem. And what we learned, which was not quite clear for us -- we kind of knew it, but not quite -- that the domain industry supply chain is deeper and messier than you expect.

When you look at it, you have one registrar you try to work with, but who this registrar really works with, you have no idea. And how many intermediaries are between that registrar and the registry, you have no idea. If you have human interaction in the chain, an error is inevitable. Plan for rollbacks at every step. Know your constraints. Registry support, algorithm. If you have any limits, you cannot work around.

Automate verification. We cannot do this 5,500 times manually. It simply doesn't work. There were scripts we had to develop to get DNSSEC correct. And maybe the hardest part is getting the organizational buy-in. The discussions you might have there in a large enterprise organization are not technical ones, but most likely political ones. And as I have so many registries and registrars sitting here, I made a wish list for people here.

Please make automation as easy and frictionless as possible. We had seen that with the Norwegian domain, they are very great at DNSSEC adoption, but simply didn't do automation with the registrar we were working together with. So we needed to move that as well to a different domain, simply because there was some

friction in the automation, which was not possible. It would be awesome if more registries would support CDS/CDNSKEY because I really have a horror of the day that our DNS provider needs to do a KSK rollover, and we need to resign every single domain. And I think that will happen.

So CDNSKEY support would be awesome. And maybe rethink the pricing for updates. As an enterprise customer, I would rather pay a higher upfront price for a domain than don't know what my pricing is at the end of the year. If every update costs something, that is something I cannot budget, I cannot ask for, and need to get approval for every single update I'm getting. This is something which is really problematic in an enterprise world. And for the registrars, please, please let us automate. That's all I ask from you. Please let us automate all the things. That would be really helpful.

This is it. If there are any more questions, I am unfortunately only here today at the meeting. So please meet me. I will also stay here. Or meet me outside later. Otherwise, you can contact me on these contact forms. I don't know if we have time for questions.

EBERHARD LISSE Unfortunately, not. We are still keeping this five-minute delay, but thanks to Jens. And last but not least, Warren will tell us a little bit about bitsquatting.

WARREN KUMARI

Hi, all. This is an update to some work which I did a few years ago, and a couple of you might have seen it during an SSAC talk before. I will go through it quickly just so we can get back to being more on schedule, but I'm assuming most people have heard this quote that an infinite number of monkeys hitting keys at random will eventually produce all the works of Shakespeare.

So what are we actually talking about in this? If you enter a domain name like twitter.com -- as you can tell, this was originally done a few years ago -- it gets stored in RAM in something like this. If one of these bits were to change from a 0 to a 1 or a 1 to a 0, you could potentially end up with another name that's also a valid domain name. But surely memory bits don't just flip every now and then. It turns out that they do. One of the common causes of this is solar radiation. Another cause is just little bits of other radiation in chip packaging. And one of the best-known cases from this is originally from 1974.

This is just when computers were becoming a lot more popular, and Intel was making this chip, which was a 4K RAM chip, and they needed to make a lot more of the ceramic packaging that the chip gets encased in. So they went off and made another factory for making the ceramic packaging on the Green River in Colorado, which was downstream from an old uranium mine. What could go wrong?

Well, the water was contaminated with little bits of uranium and uranium byproducts, and this would release sulfur particles every

now and then. This does have real-world implications. There was a Qantas flight from Singapore to Perth. And one of the bits in the air data initial reference units flipped, and the flight computer started thinking that the altitude was the angle of attack. This made it panic, and the plane pitched down at like eight and a half degrees, and 53 people ended up getting hospitalized, 14 of them had to be airlifted from the airport to the hotel.

But why would you care? Presumably, you are not a plane. But you do use the internet. So I went off, and I registered a bunch of bit flips of existing and well-known domains. So all of these domain names are a single bit flip away from another well-known and fairly obvious set of domains, right?

Like, bankofamermca.com is obviously Bank of America. Chasg.com is a single bit flip for Chase, which is a well-known US bank. Fbkdn.net is a bit flip for Facebook CDN, and a couple of ICANN ones, and some Microsoft ones, etc. So I set up a web server to capture a bunch of information, and I set up a web DNS and a web server on each of these names. And at this point, I'm guessing most people in the audience are thinking these are probably just typos, right?

If Warren found people trying to connect to, for example, Bank of America, it's probably that they typoed this. Well, that's a reasonable concern. I tried to mitigate these issues with a bunch of options. First off, I tried to choose names which are further away

on a keyboard to make it slightly less likely that people were just typoing things.

But also, when I did the research, I was mainly looking for things like expanded names. It's possible that somebody might type in microsont.com, but it's fairly unlikely that they would type in msedge.b.tlu.dl.delivery.mp.microsoft.com because chances are they're not actually typing that. Similar argument, I tried to mostly watch for things that are API or system names. No user ever goes to this name. It's just used by part of Microsoft's cloud messaging platform. I also collected what are called SNI names.

This is the name that is sent from the browser to a web server when it tries to make a connection. In this case, you see it's a real name because the TLS stack thinks that it's the right name. But somewhere in the connection, the F got flipped to a different letter. I was also watching things like full URLs. The way that you download something is that you generally click on an installer link. You don't type in an awful URL like this.

So, did I find anything that was interesting? Here is a graph of just hits to my web server. As you can see on May 14th, the numbers go up a bunch. That's because I registered a bunch of names. But if you've ever put a domain on the internet, you know a whole bunch of people connect to you and try to scrape WordPress sites, etc. I filtered those out as best as I could, and then I ended up with a graph that looks more like this.

You know, a couple of hundred hits a day, goes up and down. What's interesting is that if you look at this graph, with things going up and down, and then overlay the planetary k-index, which is a sort of estimate of the amount of cosmic rays hitting the planet at any given time, you see something that looks like a correlation. I haven't done the full math, but it looks like there's something weird and interesting happening here. So that's interesting, but does it mean anything? Well, as I say, one of the things I was collecting is expanded names.

The fourth entry on this list is Teredo. Teredo.ipv6.microsmft.com. Teredo is an IPv6 tunneling technology. If I were to set up a server on this address, everybody would -- well, not everybody, at least 719 different machines would decide to proxy all of their IPv6 traffic for me. That sounds like it could be a fun game. But that made me a little sad that things like this are still happening. I then went off and started analyzing SNI connections. So this is what the browser sends to the web server when it tries to connect. And I can see there were a bunch of browsers trying to connect and request things from me.

That's a little more worrying. There's not a huge amount I can do with that. I don't have the TLS certificates for these names. So it's a privacy leak, but it's not that big of a thing. Then I started looking through my logs for access of people trying to reach login URLs.

The first one is clearly somebody trying to log into their bank, but there's been a bit flip somewhere in RAM, and so they're trying to

reach my server instead. If I felt like it, I could put up a phishing page here, and whoever was following their login link would come to me instead. Again, what the user went to was www.bankofamerica.com, but somewhere in RAM, a bit has flipped, and they're now trying to connect to me. That seemed somewhat more worrying and would be perfect for doing phishing attacks.

Then I looked some more into JavaScript. I got a huge number of JavaScript connection attempts to me. What has happened is somebody has tried to load, in this case, Microsoft, and somewhere in RAM, a bit has flipped, and their machine is now trying to fetch the JavaScript from me.

If I were to return malicious JavaScript, this would almost definitely be running in the same origin for their browser, which means that my malicious JavaScript would run in the same web context as Microsoft, and I could ask the JavaScript to do whatever I want: send me their cookies -- for the bottom one, it's a bank -- send me their money, and things like that. That made me somewhat more worried. Bad things are happening here. But I decided I'd look through my log some more and see what else I could find. Well, a bunch of executable downloads.

Here, somebody is clearly trying to download the Windows.NET installer. They've gone to the Microsoft site, but their machine is fetching it from me. If you go to Microsoft and download an executable, presumably you're going to trust it. Windows Defender

pops up saying, "This download looks weird." You know you got it from Microsoft, so you'll just click through the warning. That seemed even worse.

Actually, I don't know if this one seems worse than the JavaScript one, but none of them seem good. Then we get onto cookies. A huge number of people decided to send me their cookies. Well, I guess more correctly, their machine decided to send me their cookies. I know the site that they're going to because I see the name they tried to connect to. I have their cookie. I could just take their cookie and play it back and impersonate them.

So this one is somebody who is trying to go to their health provider, Live Health Connection. This one is somebody who's trying to pay their rent. Oh, sorry, the other one was them paying their rent. This one is them going to Atlanta Specialized Healthcare. I have their cookie. I know where they're trying to go. I can just open that in a browser, replace the cookie that I get with this cookie, and I can be them.

Looking through my logs for, I think this was over three days, I got 136 things that were clearly login cookies of some type. They contain the word authentication or auth. They are a login cookie. So I can just go wherever they were trying to go, provide this cookie, and I am them. But for all of these things, TLS is supposed to save us, right? TLS gives us encryption. It gives us server authentication. Sadly, it turns out not.

Depending on where the bit flip happens, the connection, the SNI that's presented, will be the URL the person is going to go to. I don't have a certificate for microsoft.com, so in these cases, I would just ship back the certificate, which I do have, which is microsont.com. Those do not match, so the TLS connection shouldn't proceed. Turns out, depending on where the bit flipped in the TLS stack, the match is potentially against the flipped name. And so I ship back microsont.com, the machine has clearly accepted it because it has continued with the connection and has sent me their cookie.

So TLS doesn't save you, depending on where the bit flip happened. That seems kind of terrifying, because TLS is one of the standard security things that we all sort of believe and trust. Basically, a lot of this research made me sad and concerned. But why did I bother doing this research?

Well, we've got this new round of new gTLDs. Seeing as just bits flipping somewhere in the domain name ends badly, we know that it's going to happen a bunch more if you have access to be able to do something against the whole TLD. So maybe in the next round, we should propose that you're not allowed to register a single bit flip of an existing TLD. Sounded like a really good argument to me.

And well, I figured, before I propose this, let's see if there are any existing TLDs that are just a single bit flip away from any other TLDs. So I had a look, and all of these existing TLDs are just a single bit flip away from another TLD. About 60 of them. And one of them, well,

that was weird, one of these .bom, an existing TLD, is a single bit flip away from .com. That seems kind of terrifying, right?

If you were to go to register microsoft.bom, you would get all of the bit flips that happen in the TLD part. So that seems less than great. For the .com to .bom example, I went along and chatted with people from nic.fr, and they understood the concerns. At the moment, I believe that nic.br, the Brazil NIC, is planning on not launching .bom for open registrations because there is a real security thing here.

But what's the conclusion from all of this? I don't know. There is not a huge amount that can be done about this. This is happening on millions of people's phones. It seems to happen more in cars, looking at the domain names and some other research. It seems to happen more in cars than in other things. That initially seemed weird and sort of difficult to understand. Turns out that the infotainment system in a car sits in a really hot and uncomfortable location for a computer. Sits in the dashboard, baking in the sun, and they're generally built to a fairly low price.

So bit flips seem to happen most often in car infotainment systems. Cell phones are a very common thing as well. Turns out phones are built to a price, carried around in people's pockets, and left in the sun where it gets warm. So what's the conclusion from here? What can we do to protect against these things? I don't know. We might just be screwed. I don't know if we have time for a question or two.

EBERHARD LISSE Does DNSSEC help?

WARREN KUMARI I presented this at SSAC. There doesn't seem to be anything one can really do to mitigate this other than if phones and computers used ECC RAM, we would see less of this. But ECC RAM is really expensive, especially these days. There really doesn't seem to be anything people can do other than potentially everywhere in code where you access a domain name, make multiple copies of it, and hope that the bits only flip in one.

EBERHARD LISSE Okay, thank you, Warren. Let's give Warren a hand. So this is not an optimistic ending, but let's see. Stephen should be doing the conclusion. Please go ahead.

STEPHEN DEERHAKE If there is a theme on this one, this Tech Day, it's broken down into the sub-themes of DNSSEC. The scaling of DNSSEC deployment was a very interesting talk. With regards to the DNS abuse portion of today's Tech Day, PANDI and Identity Digital did both really interesting presentations on the abuse component and their approach to mitigating abuse. What I found interesting about PANDI's approach is that they had registrant involvement and a dual focus because they had some government regulatory stuff that they had to deal with as well. Whereas it looked to me like

Identity Digital was not dealing with any registrants at all, but just with the registrars and holding the registrar's customers at bay.

We went on from there into a DNSSEC Explorer presentation, and it kind of struck me as a kind of wayback machine approach to keeping track of things, although they were only doing it for a year. And they seemed to me to be generating all sorts of interesting statistics data. And particularly with regard to the servers per AS, I found it interesting as well. From there, we went on to DNS Explorer, which was rather interesting and I think a good use of ICANN grant money. This is a follow-on to their talk in Dublin. And it's basically an archival and diagnostic platform.

And from there, we switched over to Steve Crocker's presentation on Jake, and that is working on a framework to support multiple WHOIS policies, and I thought in a rather interesting way. And the framework for matching multiple WHOIS policies, and they actually have some code behind it, given that we had a demo. It's interesting he's partnering with TWNIC. I think it's great. If you want to see more information on that, you can go next door to him.

Then we moved on to Diego's presentation on Proxmax, which basically is a toolbox for ccTLD operators. It's an open-source VM, if I understood it correctly. And the objectives there are operational, security, and stability. There's a lot of work involved in what he's been doing. I'm not sure how much more he's got to go, but that was pretty interesting.

That was followed up by the CENTR Registry Lock Task Force presentation. And based on the polling, it looks like you got a lot more interest in implementation than before the sun came up this morning, so I congratulate you. The whole point here is to try to standardize registry lock access across ccTLDs. I assume it would also apply to gTLDs as well. And it gives complete control of the locking and unlocking information, with a good set of security guidelines to the registrant on that.

Then we had the DNSSEC at scale presentation. And there again, the goal is to simplify the DNSSEC management, and anything in that regard can only be a useful thing. And the customer example was kind of frightening, actually, having that many domains across, god only knows how many TLDs.

Lastly, we closed out today with Warren's presentation on bitsquatting. I had not been privy to this before. I thought the correlation between UV, I guess, solar, and having these bits flip was -- like, what do you do about it? And I believe Warren came to basically the same conclusion on that, except register one-bit variations if you can, and sort that all out.

Overall, I thought it was really great. I do note the irony, and I do not mean this as any shortcoming on the part of the technical staff at the back of the room, that Tech Day had technical problems. And with that, I turn it back over to Eberhard.

| | |
|---|---|
| EBERHARD LISSE | Okay, thank you very much. I've offered Navid another spot in Bali when they will be on site, then it will be much better. Thank you very much. At policy meetings, Tech Day is only two blocks. So see you next time in Bali. Don't forget your surfboards. |

**[END OF TRANSCRIPTION]**