Patrycja Wegrzynowicz: …in DNS or registrar systems. There are many security vulnerabilities that can be… There are many mistakes that we can make during software development. Also we can think about performance bottlenecks or some sort of vulnerability due to performance bottlenecks, certain characteristics be revealed that give additional hints to potential factors.

Also, we need to consider backdoors, so we should verify software provided by third party to make sure that they don't include any backdoors in such software.

What are the sources of those weaknesses? As I said before, we are only human so we do mistakes, we seem to make mistakes. This is because of a high rash of social development; it's because of lack of knowledge; high rotation of developers, also changing technologies.

The other thing is complexity of software development – change in requirements; size of codebase; growing technology stacks – we all deal with those things on a daily basis. Also we can see some bugs being resolved or simply malice or laziness.

How can we deal with such things? First and foremost, it's about software development process. We need to do more tests. We've got many good practices in software development like TDD, unit tests, integration tests and so on and so on. We need to educate

ourselves; our developers and we usually go for an independent verification, so we hire experts to do all this.

Is this enough? In the case of software development, size does matter. So concerning the size of the codebases, it might be quite difficult for a human being to grasp all the relationships and discover all the problems. Moreover when we talk about testing, we can use it to show the presence of bugs, but not the absence of it.

So the solution is simple. It's not we that should think; we have computers to do that for us. By the way, do you know the guy who said that, [Rosetent]? It's the author of (inaudible), so to many of you I suspect he is in all (inaudible). So it's a good approach because they say that a good programmer is a lazy programmer. Don't do it yourself; automate as much things as possible.

So we should use automation during our software development. First of all, we can use automatic progenerative tests to scan our application, to make sure that there is no vulnerability, especially security vulnerability in it. We can use another class of software to analyze our sources or binaries without execution. It's called static analysis. Or we can add an additional layer of the other programs to analyze the execution in the (inaudible). It's called a dynamic analysis.

And so today I would like to tell you about our solution. It's Yonita. It consists of a called Scanner that performs static verification and a web scanner which performs dynamic verification. And together they can be used to discover many potential classes of defects in software.

They can be used to discover performance anti-patterns, database problems, concurrency issues which are extremely important in today's software, as most of our software runs concurrently on multi-core processes. Security vulnerabilities, which is extremely important class of issue specifically for such a critical system like internet infrastructure. Moreover we can assess our application, whether or not it's good from an architectural point of view. And also we can discover more standard defects like non-pointer exception or (inaudible) and so on.

In the case of Web Scanner, how does it work? First it discovers the structure of web applications; it tries to do it automatically by analyzing the HTLM and Java Script. Based on this we generally test this, and to cover all security vulnerabilities, we generate specific data to try to break the application.

From the security point of view, it's very important to scan a web application for authentication, authorization and session management because this usually has a lot of problems, and input and output verification covering various injections, like script injection, OS command injection, SQL injection for a pure

(inaudible); cross site scripting, cross site request forgery and many more of that sort of attacks.

So this way you can minimize the risk that there are some hidden surprises in your web applications and specifically, the ones that can influence and have a severe impact on user data or sensitive data in your internet system. So this is like besting user interfaces and public interfaces that are exposed to the internet community.

You can verify your application from the other point of view, like the point of view of sources or binaries of your application. There is another part if you need a solution that does it. It's called Scanner. Code Scanner analyzes parts of the sources or bytecode. The sort of sophisticated call flow, data flow analysis, we try to cover all execution parts and to make it visible we use some heuristics to generate a metamodel for the program. And this metamodel is stored in an inference engine; it's a deductive database.

And then we are ready to detect defects, specifically separating defects in software. So we can analyze whether or not there is an execution path that leads to, for example, SQL injection. And so on one hand we have analysis of software; on the other we have formalization of security defects, so both parts are very important.

So this was like a virtual introduction to show you from our experience why securing the internet infrastructure, you can't

forget about software because most of defects and most of the attacks that are performed against internet infrastructure are performed against software. Thank you. Any questions? Yes?

Rod Rasmussen: So I was interested in information data on actual events, where the (inaudible) to kind of justify the little (inaudible). Also a comment that I think the biggest (inaudible) is actually the people.

Patrycja Wegrzynowicz: Yes, I agree. And there was a point about social engineering. Yeah, I agree that's why securing internet infrastructure is the easiest way is to go for social engineering. However, when you take a look at top open web application security vulnerabilities, among top security vulnerabilities there are accesses and various injections, so it's important to take care of it and to include these kinds of issues in software, specifically registry software which is quite critical that the data is stored. They use a data install plus the domain names installed.

Eberhard Lisse: Can we at least mention our names because we are transcribing this information and we can get who said what?

Rod Rasmussen: My name's Rod Rasmussen.

Patrycja Wegrzynowicz: We don't have public statistics, so we don't want to reveal those statistics at the moment about, for example, registrar systems. But we did preliminary scans on registrar systems and I will tell you a little bit more.

First it started when we did manual audits of registrar systems. It turned out that they have quite a few exercises for vulnerabilities. That's the first moment we thought it would be important to cache automatic scanning of registrar systems because basically registrar systems are the ones that are exposed to the public. Registry systems are quite closed, exposed only to the registrars.

Then we wanted to do web scanning solutions and we decided it would be the best to implement the end solution because there is no other automatic solution to do such scanning. And at the moment we are after preliminary automatic scans, also registrar systems because we don't have access to registry systems actually.

There is much more vulnerabilities that we detected from comparing to our first manual audit. But we don't want to reveal it to the public; we want the registrars first to fix the issues.

Male: This is (inaudible). So in general, are these (inaudible) other systems that aren't (inaudible)?

Patrycja Wegrzynowicz: Basically, no. The same approach can be applied to the web application. In case of registrar system or registry systems, it's because of our experience and our preliminary thoughts about internet infrastructure. That's why we applied such an approach to the registrar systems in the first place. But basically the idea applies to other web applications as well.

Eberhard Lisse: Just a quick update. Masato Minda actually got a hold of us finally. He just simply couldn't get stuff to us in time to do a remote presentation, but he will be either center or right doing the same presentation in the summer, so we're quite happy to hear that he's actually okay and hopefully we'll see him then.

So next up we have Jot Powers from PayPal who, courtesy of ISC pointed us in his direction, has some interesting things. I hope he's here because I've not seen him yet. We're missing Jot. That's no good. Okay, well then we'll skip ahead. We actually have Fujiwara from JPRS, I believe, who did manage to make it here to which we are quite happy.

Kazunori Fujiwara: Good morning. I am Kazunori Fujiwara from JPRS. Today I will talk about DNSSEC validation measurement – How to count validators. Today's contents – first, assumption; then JPRS data; results; conclusion.

First – assumption. How to detect validators – JP DS RR has been introduced in root zone. JP DNSKEY TTL is one day. Thus, DNSSEC validators and JP DNSKEY query once a day if the validators try to perform JP domain name validation every day.

Definition: Validators and Resolvers – Validadors are IP addresses which send JP DNSKEY queries seen at JP DNS servers.

Resolvers – IP addresses which send JP zone queries seen at JP DNS servers.

Diffusion rate of DNSSEC validation – Host based; the diffusion rate of DNSSEC validation may be measured by counting number of validators and counting number of resolvers. Number of hosts based – Diffusion rate of DNSSEC validation divided by number of validators/number of resolvers.

And query count based diffusion rate of DNSSEC validation. Number of queries from validators equal number of queries originated by validators. And the number of queries from all validators equal the number of queries issued by JP DNS Servers. Then query count based on (inaudible) DNSSEC validation equal the number of queries of combined data divided by number of (inaudible).

Next, JPRS data. .JP has 1.2 million registered domain names. JP DNS servers serve 1.6 billion queries per day. And JP is collecting packet captures and query logs.

There are several names and multiple operators and multiple locations and seven IP (inaudible) addresses, six IP addresses and six other addresses, and the total of 13 IP addresses.

JPRS data sets – JPRS collected two days long full capture of DNS packets around JP DS was registered in root zone. JP's DS RR was introduced into root zone at about 4:38, December 10. JPRS collected from 22:00 December 9 to 14:00 December 12, six hours before JP DS was introduced and 48 hours after JP DS was introduced. And the second data. JPRS has been collecting DNS query log from two of seven JP DNS servers for seven years. A.DNS.JP and G.DNS.JP are operated by JPRS and located in Japan, easy to collect.

Result of full packet capture – When JP DS was introduced into root, two day (55 hours) total. 1.8 million IP addresses send 3.7 billion JP queries. 3,315 IP addresses send 55, 000 JP DNSKEY queries. Seventy-five percent of DNSKEY queries came from one IP address. 5.6 of DNSKEY queries came from JRPS' monitors. And I calculated 4 time slot.

This is the result of 55 hours packet capture. Before 6h 280 validators but who is this? The JPRS equates 50 DNS monitors.

After JPDS was introducing the root, first 24 hours there were 2,400 IP addresses which sent the JPDS queries are the next 24 hours, there were 2,200 validators. And the numbers are about 1.4 million and 1.1 million. If your validators are 0.168 or 0.205 a day, validator's share of queries are 5% or 4.27%. It's the result of full packet capture.

Next, result of two of seven JP DNS servers – Querylog from [AG].DNS.JP. JPRS has been collecting querylogs from A.DNS.JP and G.DNS.JP for several years. Diffusion rate of DNSSEC/Validation may be calculated from the querylogs, but full-resolvers have cache function. JP DNSKEY TTL is one day; resolvers can choose 13 IP addresses. Then, JPRS' querylog does not contain full DNSKEY query.

How to adjust? DNSKEY queries from JPRS' test validator. How many queries JPRS' test validator send to [AG].DNS.JP. The validator sends JP zone query every day, then it sends JP DNSKEY query once a day. In the example, there are continuous six days that our query log cannot detect JP DNSKEY query from the server. I make an assumption: An IP address is a validator if it sent JP DNSKEY queries in the past seven days.

This is the result of the number of IP addresses which send JP DNSKEY queries: This same line shows real [AG], real data from we have data from [AG] query log. And this line shows JP DS in root and queries are increased about 300 or 400. And this thick

line shows adjusted value. This box shows full pocket capture data – 2,400 to 2,200. From full pocket capture, there are 2,400 or 2,200 IP addresses in both 24 hours. They are similar to the adjusted value 2,400 on December 17, seven days later from December 10. The adjustment seems to fit DNSKEY query.

And currently there are about 3,300 IP addresses which send JP queries. This chart shows numbers like the other which sends JP queries. This same line shows IP addresses, the number of IP addresses queried by the [AG] query log, and this thick line shows adjusted value. And this box shows full packet capture data. Adjusting the variable, we thought about doing 2.2 million to 3.3 million. From full packed dump, there are 1.4 and 1.1 IP addresses in a day. The adjustment doesn't fit for resolvers. I chose number of resolvers as fixed value 1.4 million weekly value.

Diffusion rate of DNSSEC validator (host based). Currently, 0.23% of IP addresses send JP DNSKEY queries. Increment before December 10 is 0.17%. It may be the other DNSSEC validators. And here you see it's similar to (inaudible) packet capture data. This works in the adjusted value; the adjusted value is similar.

This chart shows diffusion rate of DNSSEC validation query based. Two percent of queries may come from DNSSEC monitors because it came before JP DS introduction. Increment is 2 to 8 or

9. Increment is 6%. Six percent of increment may come from DNSSEC validators.

Cause of increase – 6% of queries may come from validators. A large-scale organization might support DNSSEC validation. Or some users of some large-scale organization send "JP DNSKEY" queries to their resolvers, but it cannot be identified.

Who sent JP DNSKEY queries before JP DS was introduced in root? About 900 IP addresses. Why? There are many DNSSEC monitors. JPRS operates our service's monitors. Someone sent JP DNSKEY as a trust-anchor, at least I did. IP addresses which send JP DNSKEY query before JP DS was introduced may not be real validators. Then, the increment after JP DS introduction might be real DNSSEC validators. There are 3,000 IP addresses which send JP DNSKEY periodically. The number of real validators are about 2,100 (0.17%).

Conclusion and future works. Conclusion – I tried to define diffusion rate of DNSSEC validation and I calculated diffusion rate of DNSSEC validation using JPRS' data. Number of validators seems to be increasing. There seems to be about 2,100 validators. They send 6% of queries. Part of TLD DNS servers' querylog is useful to calculate diffusion rate of DNSSEC validation.

Future work and questions. I'm planning to improving accuracy, to exclude DNSSEC monitor or users' interest. I need more data.

Let's evaluate diffusion rate of DNSSEC validation. Collecting DNS packet before and after TLDs DS introduction into root is useful, or root servers can collect complete data. May I access to another data? Then comment and questions.

[inaudible audience question]

Kazunori Fujiwara: JP and JPS is in the same zone, only one JP zone.

[inaudible audience question]

Kazunori Fujiwara: Thank you.

[inaudible audience question]

Kazunori Fujiwara: Thank you.

Male 2: (inaudible) First of all, I would like to thank you for this work. I think it's very important and setting a new precedent for others to do the same research there. You were asking for more data. This

is the right platform for (inaudible), so we have a bunch of data information is already there and we're going to collect more data as we do other events; and maybe you come talk to Warren or one of the other people that are active in (inaudible) and maybe access the data and increase the research there. Thanks very much.

Kazunori Fujiwara: Thank you.

Eberhard Lisse: Alright. Well, we found Jot who was just behind Gideon so he will be up next from PayPal. Can the people who still haven't got presentations in, can you get them fired off to Christina and myself as soon as you can so we can get them up on the website and Adobe Connect? Alright, Jot?

[break in audio]

Jot Powers: … the DNSSEC and IPV6 are going to swap at some point and at least in our corporate level they have. Any other questions?

Paul Vixie: Paul Vixie, PayPal customer. So, I want to say as the author of RFC 2671 which defined EDNS 13 years ago, 12 years ago, deployment is stuck. We got whatever we got and then at about 65% of influence it got stuck and I haven't worried about that. So when I, speaking now as ISC, heard that PayPal would be our

customer for SNS, I thought, "Well, this is great because this is going to do some EDNS things," and then of course, we started getting complaints cause you were getting complaints. And what is this terrible problem we're having with vendors who can no longer sell things to people because they can't reach our DNS and so forth.

And I was terrified because I was sure that any responsible customer-

[break in audio]

Paul Vixie: …you didn't and I want to thank you for not running down the anchor chain when EDNS started hurting you. We need a lot more companies to show the same stick-to-it-ivness as you are showing in order that we can get this thing to 90% where we marginalize to people who don't do it.

Right now the people who don't do it have an axe to grind. I want to take that away and so thank you for your persistence in the matter.

Jot Powers: Thank you. I guess my only comment there is the real key is executive backing, right? If we're losing payment and I don't have an executive behind me going, "Yeah, tough. We know we're doing DNS SEC, we know this is helping us, buying them in

smaller chunks." If I didn't have the executive sponsorship, I would have been back down the (inaudible) chain.

But we do have executives who say, "We're going to sign all email, we are going to do DNS SEC. If you want to do business with us, that's the way it's going to have to be."

And so I get backing. I am by my very nature a very… I'm not particularly conflict averse, so it helps that I can also go, "Yeah, no, we're not going to do that." But without the backing we would be screwed.

Michael Sinatra: I was way back in the corner so I decided to walk up here. Michael Sinatra, Energy Science Network, formerly UC Berkeley, When I signed Berkeley.edu a little over a year ago, I used two algorithms, mainly because I was being sort of sadistic and masochistic, I think. And that created certain queries where you can get over 4,000 bytes in response from the authoritative name servers.

We had very few problems. One thing that we'll do is we'll very much let you know where all of your unpatched QMail servers are out there where people are trying to send you mail. But beyond that – because they'll do any queries and that was the one that was 4,100 bytes so it didn't even fit in most ENS serial buffers.

There are some tricks and there are some ways of minimizing responses even with DNS SEC that we can talk about offline if you'd like to and some interesting things we learned. So obviously, we're not as big as you and we're not as big as Amazon or anything like that. There are some interesting experiences.

Peter Lauscher: I'm Peter Lauscher, ISE. Hi. We've never met until now, other than email. So Paul was saying when PayPal became a customer of ours and we went live, there was much cheering and so forth. Well, I, as Asst Admin, was having cold sweats. But in regards to what had happened, from a technical aspect what we were doing is once a month we were regenerating ZSKs. And so obviously we were going through every two weeks we were publishing side-by-side our ZSKs with the associated RR6, thus the response size below.

So we're currently considering going to what a lot of people are now doing where it's a case of we publish the ZSK side-by-side for a certain period of time and then just regenerate all the RR6 at once. So try to minimize the amount of bloat in the response sizes for, say, the intermediate period.

Obviously there are other things we are also considering about the amount of time we use a certain ZSK because we had written a lot of this infrastructure several years ago when we were still trying to figure out how long to keep the KSK, how long to keep the ZSK,

so we're trying to minimize the load as we start getting into more production environments, trying to put some realism into our procedures and so forth.
So, once again, thanks to PayPal for sticking with us. I was woken up too, so…

Eberhard Lisse: Anyone else?

Roy Arends: My name is Roy Arends. I work for Nominet which is a TLD registry and I understand that when you had about 1,100 domain names and 50 of them actually work, the other about thousand or so were basically lame delegations. Is that correct?

Jot Powers: So for the other thousand we generally had… they delegated to our servers and we had SOA, but we didn't have anything that was of any value, right? Some of them I didn't even have [MX's] in, so they had SOA but they had no internet value other than that.

Roy Arends: Okay, because what I understand is why the stuff cannot be cached is because there are no records there. And if you fix this problem using SNS from ISC, you only fix half of the problem. This is not a comment on SNS from ISC; this is a comment of trying to fix the stuff. Because we, as a top level domain, will still see the bulk of

those queries because you only fixed your problem. Resolvers will still try to get to that information; they're just not going to your machines anymore; they're going to ISC's machines. But before they actually end up there, they come to us first.

Jot Powers: Right, so actually what we do, since Mark Monitor is our registrar and I think they would have to work with people like you, is at Mark Monitor, we put the records in place that have a single A record that points to servers they have doing HTTP redirects and that's all we care about. So we do try to stop on the front end. We don't just say, "Oh, we're going to put it up at ISC." We actually try to make sure that every record is there.

Now the point I made earlier is what I can't tell is if somebody on some registrar that doesn't do strict name server checking to make sure the records exist there before, you're absolutely right. I'm never going to see that I have a problem because it may or may not ever make it to my servers.

Roy Arends: Got you. But this is exactly the reason why some registries out there are checking for proper delegation and proper propagation of the zone to secondaries of sub-domains basically or third level delegations. We don't do that because for us it just doesn't scale. I understand others do that because they have somewhat different rule set. You can, of course, check once, but five minutes after

you've checked, these records might have changed. So I don't think in general you should, as a registry, should, - well, it's up to the registry basically what they should or should not do – but we can't as a registry fix all of our registrars' or registrants' problems.

The only reason I stand up here and comment – and I love your presentation, by the way. I think it was very funny – but the only part that I have a problem with is that you fixed this problem only on one side and didn't fix the real problem, which is the delegation part.

Jot Powers: Yeah, I mean I'm open to suggestions on how we ought to do that. I mean, there are a couple pieces that I think about there. When you start talking about the registry validation, I see some of that. Some of it is, especially for the ccTLDs, some of them have competing requirements.

Like I think .nl has to have only three name servers listed; whereas others want no more than two. So if I have one domain that needs to go multiple places, I can't use the same infrastructure everywhere and I want to stamp stuff out. My goal is not to spend all my time doing DNS administration, it's to try to process payments and keep the site up.

So when I see those competing things, you're right. And there are a lot of times where we get that, "Well, you have missed some

arcane little bit." I really don't care; I know my stuff is accurate. But you're right. Validation, I think, is probably of limited value. Thank you, everyone.

Eberhard Lisse: Thank you, Jot. Alright, moving on. Next up we have Bart Gijsen.

Bart Gijsen: So I guess everybody can hear me right loud? Welcome. Good afternoon. I think we slid into the afternoon session already. I'll be presenting to you regarding our research on DNS, DNS (SEC) client analysis. My name is Bart Gijsen. I work for TNO. TNO is a research institute in the Netherlands. And the work that I'm presenting, we did together with SIDN, the registry for .NL. We got some assistance from our colleagues at the Laboratory for Quality Software in Eindhoven and we used some tooling from [NL-net], so it's not really only us doing this kind of research.

Before I start, I think the research that we did so far has been pretty self-contained. But the reason that I'm presenting it to you is that I'd really like to have some feedback and if you have any experience on client analysis DNS or DNSSEC client analysis, I'd be really interested in hearing it from you. We are actually at a junction point where we're going to head our arrows to some direction and we'd really like to have your feedback on what most beneficial direction would be.

Just to give you a brief background on why are we looking at clients? The bulk of the DNS traffic analysis that's been done so far is, I guess, on the authoritative side, on the resolver side. But, as mentioned by one of the speakers earlier, it's very hard to look at what's behind this resolve. What are these clients actually doing? That was one of our particular reasons to dive into the client behavior a bit more.

What you see here is a pictogram with several kinds of literature. It's not an extensive list, but it also shows that most of the traffic analysis has been done on the right-hand side of the picture. So the key question that we posed ourselves for is how will DNSSEC change the behavior of DNS client querying? And in particular, as mentioned also by the previous speaker, what are, for example, response sizes larger than 512 bytes going to do? What if validation errors happen and the resolver starts sending servfails back to the client? How do these clients – and I'm now talking about your laptops and PCs – how are they going to react to that?

The basic outline of my presentation is that we did some experiments with DNS clients and I'll then skip to telling you something about the impact of that and I'll wrap up with a summary and the next steps.

So what experiments did we do? On the left-hand side is an experimental set-up that we used. We used several clients, placed a DNS resolver in between. Actually, we had a Bind 9 resolver in

between. And then we had a controlled DNS server. So this was a fully contained, not internet hooked up experimental set-up.

For the client's side, we used several operating systems such as Windows XP, Windows 7, Ubuntu Linux and Mac OSX and we used several browsers on top of those. We didn't really do all the combinations of browsers and operating systems; we did quite a bunch.

Unless stated differently, we left the settings as much as possible on the default, so we simply downloaded all these versions and then started experimenting with those default settings.

Then what we did was we defined the affirmative side, and as I mentioned, we had a controlled DNS server there. Actually, we used the LDNS tool from [NL-net], and we configured a domain name which would give us a servfail response which would give us a valid response. So we had one DNS domain name for each type of response that we'd like to check. Okay?

So then we could query for that specific domain name and see what the reaction of the client would be in case of a valid response, in case of a valid response with a size larger than 512 bytes, in case of a non-existing domain, whatever.

And we did that by using ping in order to see what the operating system behavior is and we did that *via* browser. And the browser,

of course, gives us the characteristics for the browser itself and the operating system underneath.

So if you do a lot of runs with that and you're looking at what is the client behavior in terms of the query rates and the query delays? And you get screen shots such as these and I'm afraid it's a bit poor readable, but I'll explain it to you. It's a TSP dump for one of the examples.

Here we see a TSP dump for a Linux Ubuntu with a Firefox browser on top of that. And we directed a query towards the domain name or the IP address that will send back a servfail response. So you regress for the core A record and the Linux gets back a servfail and what it does is it does three immediate retries for the query record and gets back the same servfail simply because we configured it like that.

What's not very strange in this kind of behavior is that it retries. Why does it retry three times and in particular, why does it retry immediate? Okay? Then often not getting any response or actually its default behavior and then queries for the A record and we see exactly the same behavior. It receives the servfail and then they're full of three immediate retries for the A record.

Then it decides, "Okay, I cannot get a valid response here, so I send it back to my browser," and Firefox requests the operating

system to do all of that once again, just to verify. So you end up with 16 queries in 0.14 seconds.

Well, if you do that for different types of response types, you get tables like this; actually the TSP dump version that I've shown you previously is in the top table on the third row where you see the servfails and Firefox making the repeats a factor of 2, right? The Linux operating system can be attributed to doing 4 HYs, so 2x4=8 queries in total. If you do that for other kinds of responses, for example an [Ulnix] system domain; with the Linux system and Firefox, you would get a 4 times repeated query.

Keep in mind that we configured a third-party site and the client to only go to one primary DNS server. We disabled going to a secondary DNS server. If you do that, you get a doubling of this once more.

More or less the same behavior we see on the Mac OSX operating system; whereas the default Safari browser on top of that does have some different behavior than the Firefox on the Linux machine. In total you see this kind of repeated behavior from clients like Linux and Mac OSX.

As I just mentioned, we configured this to go through the DA single sort of name server, so only a primary. It will be doubled if you go to a primary and a secondary. And this table that I've shown you here is only for getting a single record so any record a

or a [AAAA] record - if you do it for a [AAAA] record, you would have to double it again.

Now we were quite interested in how our good old Windows system's gonna behave in this kind of experiment set-up. And we were slightly surprised by the outcome. In fact, first of all, if you're running Internet Explorer, but also other types of browsers on top of Windows, we don't see the doubling.

There seem to be different versions of Firefox, for example, for Windows than there are on the Linux systems. And in particular also, in case of a servfail, we don't see any repeats. It's actually immediately fed back to the browser and the browser feeds it back to the client. I'm not saying that this is better or worse than the other; it's simply surprising to us that it's quite different actually.

But keeping this in mind, having those immediate responses, we were triggered on where they would go. And actually with the Windows systems, we've observed that it doesn't really matter what kind of browser is on top of the Windows system, it doesn't do any amplification of the querying and response, irregard of the type of response.

Not really investigated in our research, but we've observed several other types of sources for more or less aggressive DNS behavior, such as greedy apps, the bonjour protocol, Facebook apps – which are doing a lot of synchronization which generate a lot of DNS

queries – and also features such as pre-fetching, of course, which use a lot of DNS traffic and from the client's side.

It's interesting to compare the impact of this kind of aggressive behavior to the sender behavior of the client in case of servfail and in case of [serverils] in the case of Linux system domains. Once again, we didn't really dive into these features too much.

Having said so, of course the client here can to some extent be shielded by the DNS resolver in between. And, indeed, if we look in our experiments we included a DNS resolver, a BIND DNS resolver, in between and we see some dumping of this more aggressive client behavior by the resolver.

For example, in case of no replies, when you ultimately end up with a time out, in that case, the resolver will retry seven times with exponential timer backups towards the authoritative side while holding back all of the retries that the client does.

Similarly, of course, the resolver, if configured correctly, will cache positively and negatively so we will cache the developed responses in the non-existing domains. And also the truncation, it will help you there to make this client behavior a bit less aggressive.

On the other hand, it also does some amplification, and in particular, if you look at the servfail responses, it does a double

check. So there's another factor of 2 in some cases where you would get back to your servfails.

What we also looked at is how does it handle unvalidatable responses. And what it actually does is it returns unvalidatable responses servfails, which is more or less correct behavior.

But it makes sure that the client will trigger on this. And in particular this Linux and Mac OSX, they will really start to react with their immediate responses to this servfail and we see a lot of validation errors being generated actually. So it does contribute to some of the issues and some of the larger volumes of servfails that's been reported by earlier presenters.

Then we try to figure out where is this - can we get a solution for this? Do we need a solution for this is another question. And we asked our friends from the Laboratory for Quality Software to do a code analysis, and I think this is more or less the same type of code analysis that's been presented by Patrycja, I think, based on a static code analysis and a dynamic code analysis.

Given the fact that we observed particular aggressive behavior in the Linux Ubuntu version and in the Mac OSX where our first ideas were, "Okay, it should be in the glibc somewhere. It's the system library that does the DNS querying."

So we requested them, "Could you do a code scan of the glibc?" And their observation was, well, first of all the system library as a whole looks pretty good actually. There are no extraordinary characteristics found in the library as a whole.

But they were able to find, to pinpoint the code part which is responsible for this replication in case of, for example, servfails and non-existing domains. They tried to figure out, could you come up with an alternative to make sure that this behavior is canceled. But they found out that the code part in particular that's responsible for this replication is rather complex.

So we said, okay, let's not make our life too difficult. There are experts in this field which would know much better than us how to make any improvement or just any change to this part of the code. So let's leave it at that and let's first start to look at what's the impact of this. I mean, we really didn't address that question at this point in time.

So that's where we started to come up with a positive impact model just to try to get some figure on how much traffic is this behavior generating extra. And we started out with some model that isn't really doing something like perfect behavior.

Our idea was we were going to put this characteristics, this client behavior in this model and compare the DNS volumes that there will be between the nicer or less aggressive behavior of the

Windows systems comparative to the Linux operating and the Mac operating systems.

And then in this model we also took into account, of course, that the bulk of the traffic will not be generated by Mac OSX – but my grandma uses Windows, right? – so we took this rating parameters into account in this model. I'll not be discussing this model in detail, but I'll just show you some results.

Here's the first thing that we found out. Same picture again, with on the left, the client side, in between resolver and on the right hand side, the authoritative side. And there's these red percentages. What they represent is the following:

We started off with the base case where we have the perfect behavior of volumes with the normal rates between Windows systems, Linux and Mac OSX. And then we said, "Okay, let's suppose that we are able to find the cause of this aggressive behavior and neutralize it. So let's suppose that all world would consist of, say, the Windows kind of behavior systems. How much less traffic would that end up with on each of the regions here in the DNS system as a whole?"

Then what you see is that that kind of improvement doesn't really bring you too much; it's only in the area of a few percent of less DNS traffic. And why is that? Of course, I think it's quite logical – first of all the more aggressive behavior in the Linux and Mac

OSX – they simply have a relatively low share of the DNS clients and secondly, this aggressive behavior only occurs in case of, say, exceptional responses. Servfails, non-existent domains in the lower parts of the authoritative side are the minority of the response types.

Nevertheless, you see some difference here. In general, on average the behavior is not really devastating. There are particular cases where it's becoming a bit more interesting. For example, suppose that there is some secondary level domain owner which is making some errors in their DNS SEC configuration, which may lead to validation errors and resolver. And then you get the Linux and Mac clients kicking in and displaying their aggressive behavior.

Now suppose that - I think we've run this scenario – suppose that 10% of the DNS would ultimately end up with a validation error, then you get this kind of numbers. And, of course, this will blow up, according to this input parameter, where you say, okay, if none of the responses for a second level domain would be validatable, you'd blow this system up as a whole.

So this just shows you that in particular cases this aggressive behavior will contribute to making it more aggressive. You'll see this proportionally much servfails in your domain if you're making errors in your validation. Okay?

So that's about what I'd like to tell you and once again, I'd be very open to any contributions and experience that you've been having on DNS client here. As far as we've been seeing, we've been seeing that the Linux and Mac clients display more aggressive DNS behavior than we see on the Windows systems, which is partly dampened by the resolver, but partly amplified.

The impact on the average DNS traffic is relatively low, but keep in mind that it's fractional, right? It's yet another few percent of traffic that you wouldn't like to have. And in particular for some bad cases, this kind of behavior will not help you; it will really make your problem worse.

The next steps that we are planning is to share experiences with other experts. Once again, we are very open to that. In particular, we are also looking at can we have any kind of contribution to improving glibc. And then do some further analysis of quantitative scenarios. We are currently using data from Surf Net, one of the smaller ISPs in the Netherlands to verify this kind of aggressive behavior and indeed, in the logs we do see those immediate retries.

And if you fingerprint what kind of clients they would come from, it does seem to be like the Linux clients which are really being seen in the traffic of this ISG. We're also working together with ISDN to see whether we see this kind of behavior all the way up to the authoritative side.

And the question that's been nagging us so far is this aggressive behavior, the impact is relatively low, but what if mobile internet really blows off further than this and we have a market share there of Linux-based systems which is way larger than we have currently in our PC market. Where would that take us? So, that's about it. Any questions here?

George Michaelson: Hi, George again, sorry. You're familiar with the Happy Eyeballs Draft, aren't you?

Bart Gijsen: Yes.

George Michaelson: Because some of the behavior that is deployed in existing code are attempts at various things that go in front of Happy Eyeballs to do thread to DNS races to see which query comes back first and make a decision what to do. If you can characterize that behavior, that would be enormously useful. Very, very interesting.

Bart Gijsen: Quick response to that – it's a good observation. There's several kinds of additional factors that will make sure the amplification's right. And that's indeed one of the things that in our quantitative modeling we'd like to have some idea about what does this effect have relative to the DND behavior that we've seen.

But also there's a domain incompletion. There's all kinds of other stuff and what we are trying to do with this quantitative scenarios is try to quantify how bad is one behavior relative to the others, such that we can really focus on the most important issues that are causing all kinds of aggressive behavior. Thank you.

Russ Mundy: Russ Mundy from Sparta. We've done a bunch of work in instrumenting applications with DNS SEC validation for the last number of years, Firefox being the most obvious one. And, oh, my good news is it's not running on Windows. Yay! That's been very, very hard to get done.

But over the period of time, probably the last four or five years that we've been doing this work, and (inaudible) the validation capability into the web browsers, we have observed – not studied or don't have any real numbers on it – but we have observed a very significant increase in techniques being used by browsers for, George, it's the Happy Eyeball idea – doing everything they possibly can do to get as fast DNS resolution in front of the user as they can.

And the results of that that we've seen in terms of our fully validating browser is the more stuff that shows up on pages and when you look at the last thing - I think we did as far as collected the number of queries and responses to fill up a browser with CNN

home page, just as a commonly well-used starting point – it was something over 100 queries and responses to start out cold for one page.

And so my point is it's only going to get worse. Now, what the actual analysis of that and the real world impact is, I don't believe anybody really knows yet. But studies of this nature are incredibly useful in trying to help people get some insight forward.

One thing that we've been able to see in our browser work is that with the separation of the recursion and validation done on the client - and we just started to look at the impact regarding recursion on a recursive server on a separate machine – we think there may be some very good games that can be done there that'll let the end users still do validation, but not have the heavy recursion mode. Thanks.

Paul Wouters: I'm Paul Wouters from Excellence Corporation. I'm also one of the Fedora developers with respect to DNS packages. For Fedora we're looking at actually installing a resolver on every client for default, but there are some technical issues that prevent us from doing this right now.

One of them is that people prefer to use the Bind name server because they know it best, and Bind cannot take a dynamic (inaudible)forward, so if you change the networks from one day to

PE network to another, you would like to dynamically be able to change the forwarder so you can at least use the caching infrastructure of your network and not become like your own grouped resolver and adding even more queries to the network instead of reducing them. So that's one of the things we're looking at. And I would actually like to talk to some of the IC people to see if we can do something. Unbound does support it, but the people are reluctant to switch.

Bart Gijsen: I think to add to this behavior that we've seen on the Bind version that we use, I guess there will be the features to configure that cancel this behavior more. But the standard behavior and for an inbound resolver is if you have a validation error, then keep that in mind for 60 seconds and don't send any of the new queries from the clients to the (inaudible) side is going to be very effective as a problem to resolve validation errors. I guess that kind of technology should be used a bit more.

Ray Adams: Hi, my name is Ray Adams from Nominet. First off, I think this is fantastic work. This is really what I'd like to see a DNS OR CCNSO Tech Day. I think this is fantastic work.

We at Nominet, we only have the ability to look at traffic from a resolver towards our authoritive name server sets. And you guys

have done this even without source data; you've created this in your own lab environment.

Two questions on this – Is there a paper associated with this work and if so, can we get it here or I'll have to get it.

Bart Gijsen: It will be soon. I'm sorry. It is not out yet.

Ray Adams: No, that's fine. I'm looking forward to seeing it. You mentioned other platforms, like mobile platforms. And I'm particularly interested in IOS, and then, of course, the android behavior as well. Is there any further research planned on this? Are you guys willing to work on this?

Bart Gijsen: Yes, exactly, the bottom line. During doing this research together for example Antoine here from ICDN, we identified that this may be more interesting from the mobile platforms. So far the research has been contained to the more like laptop operating systems. But we're really eager to expanding this to the mobile platform.

We did have one chap with a larger ISP in the Netherlands which provided us with some data in particular regarding mobile traffic, and they said so far we don't really see this problem yet because they shield a lot of it. But that, of course, they have no idea where

it's coming from; they have no idea where their android systems, for example, are really that Linux-based that we see the same kind of behavior. But, yes, we would really like to dive into that and once again, I'd like to invite any other experts here to contribute to research like that.

Eberhard Lisse: Thank you very much. Alright, next up and last up before our lunchtime break, we have Olafur Gudmunsson from Shinkuro and we'll get him up and going.

Olafur Gudmunsson: As I'm the only one that is standing between you and your lunch, I'm going to try to be quick and entertaining. Okay, so the first slide is what do we know about DNS resolvers and in this context I'm talking about recursive resolvers. Go back to the slide before please. There are certain questions that I wanted to be able to answer. And if you ask somebody about it, yes, some of these are economical; some of these are technical and some of these are standard compliance.

I care about standards because I happen to be Chair of the DNS Extensions Working Group. I worked on DNS Extensions for a long time and I do research and consulting in DNS. So here are some of the questions. Next slide please.

And I'm going to start by showing you a few test cases that I have seen over the times and I'm going to talk a little bit about them and then bring everything together at the end.

This is something that happened about a year and a half ago. I'm not going to name any names, but this is what happened. The vendor contacted me and said, "Do you know how resolver X behaves when it comes to sending queries around?" I didn't know. I was thinking about it.

So because the vendor had got a call from the operator and saying, "You're overcharging us." And when they said, "No, we're not. That is what you're stealing traffic." So the final question was, was the model they used was wrong. Well, what was the model based on? Next one.

I've been doing the research on what will make transfers of DNS domain from one operator to another operator not work or fail. And as looking at the fundamentals of it, there is one thing that sticks out like a sore thumb. There are two NS sets – one of them comes from the parent; one comes from the child.

According to the specifications, the parent is authoritative for the existence of the NS set. The child is authoritative for the contents of the NS set. This was negotiated, this wording with me and Paul Vixie and Paul (inaudible) many, many years ago. But the

question is which NS set do resolvers actually use? This dictates where traffic goes, for example, if the NS sets are not the same.

And then there is the question of what if the resolver is using the NS set from the child and it asks the child server frequently enough it keeps getting NS sets in the authority section. What happens if it is just refreshing the TTL? Then it may get stuck on the child server. And if the domain is delegated, the old operator does not turn off service, then people with this kind of a resolver will get stuck on it forever, or until they reboot.

So next one, please. Okay, we are talking about various aliasing in the IETF. And one of the questions that came up – what percentage of resolvers supports DNAME? Well, there are two ways to ask this question – what percent of implementations; what percentage of traffic?

Well, we were able to answer partially by looking at what certain implementations do, but we have no idea what is the market share. What is a market share for (inaudible), for example? What is the market share for power DNS recursion? We don't know. Does it depend? And what about all the other resolvers that are out there?

There are lots of little resolvers. I'm trying to figure out, like my colleague from Japan, how much the NS set validation is going out there. I'm looking at traces from a global domain called .org. .Org is a little bit different from .jp; the DNSKEY TTL is 15 minutes.

The DSs they give out is one day. I've been looking at 15 minute traces. I'm looking for both the DNSKEY and DS queries because DS queries are probably a better indicator of whether somebody is doing a validation than the DNSKEY, because a bunch of key monitoring tools are asking for DNSKEY queries. And if you ask for a DS that probably means you're guarantee.

So we have to look to check if somebody's actually doing validation, we have to look at DNSKEY over time and see how far they're spread apart. Go on.

Well, there is a problem. If you look at the NS set for .org, there are six names in there. Afilias – the operator for .org operates 2/3 of the NS records. An outside vendor, Packet Clearinghouse operates 1/3. Packet Clearinghouse has more sites than Afilias does and theirs is more spread out over the world.

I see that about 50% of the traffic in the traces I have and I can't get the other traces because in some places we're talking Clearinghouse has servers; there's no way to get the data back.

Simple question – what's the probability of seeing the queries that I'm interested in? And then it is what kind of resolvers am I seeing? And I started thinking, "Does it make a difference whether a resolver is busy or whether it is sporadic?" Does the resolvers

for Comcast behave differently than the resolver I have in my basement? Next one.

Similar questions – Whenever I explain to people that there might be child sticky resolvers out there, the answer was always, "No, they don't exist." And I said, "Well, they do." And they said, "Prove it." Okay? Then it is what is the percentage? That's typical to have.

So I decided to experiment to see if I could discover what is what. One way was to set up a zone where the parent and child set differ slightly. Doesn't really work unless you have identical probability of questions coming, but we don't know if that's true. So I set up a zone which has only one name server on it. And here's the set up in the parent.

We have a parent and it points to one of the named servers for the child. And that gives out an answer called red. And in the NS set this one gives out, it points to another name server and if you go through that one you get a different slightly answer. And when I'm checking for transfers, then I keep flipping between the blue and reds. This tells me whether the resolvers are picking it up.

But how do I test? So I thought, "We have a bunch of these open recursive resolvers out there that people are complaining about. But are they a reasonable sample of resolvers that people use?" Well, let's take a look at what they do. So for my experiment, I set

it up with the NS records up to 17 seconds; for TTL text records for seven seconds and at the end of my run, I asked them what the version was, just to hope they would tell it.

These low TTLs may be skewing the results. We'll come to that later. But before you run an experiment, you should know what you're looking for. If I have basically three different types of resolvers that are classified in my model, there is the parent centric servers and they always answer from the NS set that is coming from the parent. So you should see a string that is all ps from them.

From the child centric sticky – they go to the parent in the beginning for the first query because they see the query as soon as they get to them and they learn about the NS but they keep the data for a little while and the child centric, it will go yo-yoing back and forth.

Okay, next slide. I got lots of different patterns. Here are the top ones. Well, you see the child centric, which some people believe is the right way to do it is dominant. Child sticky – they're 15% of the whole traffic. That was more than I expected.

If you go all the way down, then you see there is the parent centric show up in 9th place. But if you look at the other patterns, they look like they are some variations of being child centric, but there are some minor variations. I don't know why I have this long "C"

string here; that should never happen. It should never have happened that way.

So these are some of the statistics. There are 10% that I'm not sure what is happening. But there may be other reasons why this is happening. They may be having different rules about how they are handing the TTLs. For example, some implementations have a minimum TTL that they will cast things that, if something comes in with a 30 second TTL, they will still cast it for 60 seconds.

There may be ancillary use. I talked to an (inaudible) vendor of a resolver who implements an (inaudible) and if he gets a question, that's identical, he just changes the fields and the headers and reuses the query. I may be running into an (inaudible) server, so I switched in the middle of the run from one to another.

There may be forwarders in front of the servers that I'm talking to and I'm seeing the behavior of the forwarder, not of the recursive resolver. So there are all kinds of strange configurations that our people are using.

Okay, does behavior depend on implementations? The most common implementation that I saw from the version string was by Bind 9.6, which is good news in a way because it means it's modern; it's a bad thing because it's not the most up to date. If you look at Bind releases, Bind 9.3 and later are all child centric,

according to the source codes I compelled on my machines and tested in my laboratory. 9.2 and older are child sticky.

But among the Bind implementations, let's say that they are child sticky. Only seven of them say that they are 9.2 low. How come? If you look at Bind 9.6, which is the most common, there are only 12 out of 62 that behave like the one behaved in my lab, and even after I relax it for slight timings, there are still behavior ends there that are not explainable.

For example, look at the bottom one. This shouldn't be happening. The good news is there's nobody said they were Bind 4; they might be lying. They might be saying they're Bind 9.6, but they are something else. So, next one.

But here are some concerns. Two very popular implementations off resolvers are parent centric. Maybe this is because my experiment was over such a short interval and with such a low TTLs that their minimum TTL policies overwrote the behavior that I expected, so I would have to run the experiments with longer TTLs. DNS [cas] and open DNS are child sticky, possibly same reason, but maybe not. Actually in the case of a DNS [cas], I have confirmed it is completely child sticky.

So, here are some of the other reasons why things may be behaving differently. Next one. Okay, next question. How does a resolver scatter queries? Well, when the resolver is discovering a domain,

here's what it's supposed to do. It asks one name server out of the DNSSEC a question. The next question goes to a different one until it's gone through all of them once or multiple times and figured out where the traffic should go. They can do this forever.

They can also create what is called an RTT band. They will talk to the name servers that are close to them, *i.e.*, answer with a short. How big this RTT is depends on them.

So we don't know how often they do this and how often do they forget about all the servers they have and to start this learning process again. And when we are looking at traces like 4JP and (inaudible) and others, it is important to know if you're only looking at a subset is, what is the probability occurs are coming from a busy one versus a sporadic one?

And being a busy in one location can be a sporadic in another one. For example, I don't expect ISPs in Brazil to be asking a lot of questions in the Czech Republic. I could be wrong, but, or the vice versa. And one thing that I've seen from looking at all these traces is address does not equal resolver.

There are lots and lots of cases where there are multiple resolvers behind the (inaudible). So, I was looking, why am I getting a DNSKEY query basically like a clockwork every five minutes from an address?

Well, it's a very busy resolver. No, there are three resolvers behind one address. So, we have to keep that in mind. And how to analyze the data to figure out how many resolvers are sitting behind an address is hard. And if we start seeing more people starting to put resolvers on the end systems, this is going to get even worse.

And if we want to talk about DNS validation, we have to be able… one DNS SEC validator can label a whole address to be DNS SEC validating, even though it is only sitting on one laptop in one corporate environment, if that's a busy enough laptop.

So, if I'm looking at the [org] data, for a sporadic resolver I have a high probability of seeing the regularly schedule DNSKEY query, so looking at a 15-minute sample, I should see it. But for the busy ones, there is no way I'm going to see some of them because they have homed in on sites. Like because of where Afilias' name servers are versus where the Packet Clearinghouse ones are, I'm not going to see it.

For example, South American – I'm probably not going to see it at all, similarly. The other question I would really be able to start analyzing and is of great interest to me is how will DNSSEC update differ in different regions? So being able to associate DNS query with the region is important, but can we do that with all of these strange resolver behaviors people are doing?

So it's not easy to look at traces and figuring out what's going on. What we really, really need is to be able to start thinking about building models that people can re-use to figure out what is going on. And if I have all these traces from, let's say, .uk, for example, and I only see two of their servers, can I draw any conclusions? Other thing is we don't need to see all the queries; we only need to see certain samples.

For DNSSEC validation, just looking at DNS queries is sufficient, but that does not tell us how many queries are issued by others and we need to weigh them. And then, of course, resolvers change over time.

One of the questions that people always ask me when they have a problem is, "What does Bind do?" I'm now very happy to answer to them and ask them back, "Which version of Bind?" They have changed so much over time. Power DNS has changed its behavior multiple times. I don't know how many times Nominet has changed their behavior; Windows has changed their behavior.

For example, last week Bind 9.8 announced they changed their RTT behavior. Whether it is for the better or worse, we don't know. Next slide.

But there are things we can look at and some of these RTT behaviors were originally designed at a time when we didn't know the world. They were designed before we had [Anicas] servers all

over the world. So maybe that has to be revisited. Thank you. Questions?

Paul: Paul (inaudible). Did you do these tests with both an unsigned and a signed domain to see any difference between whether you would get less resolvers stuck on the child when they needed to validate a DS record?

Olafur Gudmunsson: No, because I have no control over these servers and such a small set of them claim to support modern DNSSEC. I don't care, necessarily, in this example about what a DNSSEC capable resolvers do; I wanted to know what do resolvers in general or bigger sample does. So, no, I did not look at it, but I could. Or you can run the experiment.

Eberhard Lisse: Lunch! Anyway, thanks for coming out this morning, guys. We're going to break until 2:00 for lunch and then we'll get back here and get back at it.

[break]

Ondrej Sury: Hi, my name is Ondrej Sury and I am from CZ Nic and today I'm just more like a messenger because the work was done by my

colleague, Karel Slany who couldn't be with us because he was ill. Next slide.

So the motivation for doing work on accelerating DNSSEC on GPUs was like, well, there is a lot of work to do and there's large scale data sets and it's common today that it's good to process larger data sets to GPUs, so we thought, hey, we could try that. And the speed for it typically relies on the design of the parallel, not the speed of the GPU. So we tried it and here's the result.

This is just a graph and trends. I think that Nvidia just published a new platform which has more cores and some more power. Next slide, please.

The GPU is a kind of special architecture so they could afford to put more power to derail competition. But the control logic is reduced and there is only a limited flow of the code and the program models are quite unusual. While you must think not in the usual way when you code for a CPU, so you need to revoke the algorithms. And so the GPU is not real multithreaded processor so the approach is different.

The pros is that performance can be very high and the CPU can do other stuff when you compute on GPUs. The cons is that it's very hard to achieve the peak performance, and there is high latency when you need to put data on the GPU. And you also cannot

communicate between the threads on the cores because it really slows down the whole thing.

So we implemented RSA on GPUs and there are some requirements which needed to be done before it started working. We needed to reduce the code diversions, which means branching in the code and all cores process the same execution path.

There's no communication between threads because the synchronization slows down the code and we needed to reduce the expense of arithmetical operations. So right now we use only the addition and multiplication because, for example, division is very expensive on GPUs.

Now is where even I start to get confused, so if you have any questions about implementation, you really need to ask Karel. The implementation uses the Montgomery exponentiation algorithm in a residue number system, whatever it means. And it's based on a work of Kawamura and it's Cox-Rower architecture. There are references at the end of the presentation so you can read the original papers if you want to.

And it's based on modular arithmetics and the size of the model is limited by the width of the arithmetic unit. And you need to properly choose the Residue Number System base and it allows you to replace the division by only multiplication and addition.

So the implementation details – the encryption process uses the register-width numbers and there are no dependencies between operations like carry between the numbers.

The arithmetic operation where is used only to addition and multiplication and there are some pre-computed values which are key dependent, so you need to compile the code if you change the code and upload it to the GPU. The code path is key-dependent so you cannot generally do any key at any time. You need to actually upload the code to the GPU if you change the key.

And all GPU cores execute the same code. This means that the performance is increased; but on the other hand, you need to sign many things at the same time to be real effective.

We already have the library and we have working implementation of RSA1024. There is also experimental support for bigger sizes of the algorithm, and that was not yet properly tested because you need to change the library when you change the size of the RSA. And the build process requires Nvidia SDK but for the runtime you just need the libraries.

Here are the numbers. So you can see that the numbers, we are using the Nvidia GeForce GTX 480 and it has 480 cores. There's a note – they just released the new hardware which has even more cores at faster speed. And compares to OpenSSL implementation on single core Intel processor and the gain is 3.5. But that's gain

against the single core, so if you deploy more cores in the processor, it will be probably faster.

So if you have any questions – and try not to be hard on me – then you may ask. If you have some more complicated questions, better send them to Karel and is email address is on the first slide.

Roy Arends: Hi, this is Roy Arends, Nominet. Not so much a technical question, but an overall question. Why? Why do RSA do a GPU? For instance, about two years ago, little bit over two years ago, we at Nominet did some simple research on how to speed up things and we came to 40K signatures a second – ours was then 24 – on a device that was twice as expensive as this GPU you just mentioned.

So you have a speed of about 800% with a price tag that's about double this. So I understand it's cute to do, but I don't see any value in this in the production environment. So maybe you can enlighten me.

Ondrej Sury: The answer is because we can. No, really, we are the research lab so we ought to test new technologies and things and, well, we thought that it might be the way. And it seems like we cannot achieve higher speeds using the GPUs unless something queer happens, but it was an interesting project.

Male 3: I have a question. Does it make any pictures?

Eberhard Lisse: Thanks for filling in. Alright, next up we have Fugiwara again.

Kazunori Fujiwara: Good afternoon. I'm Kazunori Fujiwara again. This time I will talk about JPRS' DNS server/service evaluation – user side evaluation.

Contents – I will introduce how JPRS' DNS server/service is evaluated before the DNS software/service will be used as JP DNS servers.

Motivation, Evaluation, Result. Motivation – TLD DNS servers must always answer correct DNS responses. JP zone is a complex zone compared to gTLDs and root zones. Because of this complexity, JPRS is more heavily affected by DNS software bugs than other organizations. Then, JPRS evaluates DNS server software/service extremely deeply before using them as JP DNS server.

JP zone's characteristics – JP domain name structure consists of multiple type of domains. General use domain name: it's second level domains: like gTLD. Organizational domain name – it's

third level - .jp, jprs, .jp and so on. Geographic domain names – it's third or fourth level domains – metro.tokyo.jp, or city.chiyoda.tokyo.jp, pref.nara.jp and city.nara.nara.jp.
JP zone is one zone. No delegations on co.jp or ad.jp or Tokyo.jp. There are many empty non-terminals.

It is JP zone example – It is jprs.jp the origin and it is jprs.co.jp the original. And it is city.chiyoda.tokyo.jp delegation and all delegations are in one zone.

JP zone's update – JP DNS server uses both AXFR and IXFR to transfer JP zone – AXFR once a day – useful for changing DNSSEC parameters; to avoid possible IXFR bugs, but JP did not confront yet. IXFR – normal update, every 15 minutes.

Evaluation History – When JPRS had chosen secondary DNS service; when JPRS introduced DNSSEC. In that case, Bind 9.4.3 to 9.7.1; DNSSEC evaluation itself was another work.

Version up of DNS server software – Bind 9.7.1 to 9.7.3 and planned. When JPRS will use another DNS server software: Bind 10 or NSD or another software.

Evaluation steps – 1) Define current running software as a reference; 2) Read new software documents carefully; 3) Use the target software for small zones; 4) Perform zone transfer test; 5)

Perform DNS response performance test; 6) Perform DNS response test.

Define reference version – Writing a reference DNS response generator is best solution, but it is hard and comparison with current running version seems to be useful. When JPRS has chosen secondary DNS service, current running DNS server as a reference – Bind 9.4.3 or 9.7.1 was a reference. When JPRS introduced DNSSEC, Bind 9.4.3 was a reference. Version up case reference was 9.7.1.

Read documents carefully – It's obvious – changes, manuals tell us a lot of information – Bind 9's changes may contain important bug fixes. After a new version released, security advisories were sometimes open to the public. Read with extra caution by noting the following points: differences from reference DNS server; changes of default settings and paths; changes of configuration syntax; bugs or fixes after the reference version released.

Use the target software for small zones – to collect operational practices. I used the new version on JPRS' lab network and my private environment.

Zone transfer test – set up the test target as JP slave server – both IXFR and AXFR will be performed. Test tools sends JP SOA query every second to the master and targets, collects and parses

responses. After zone data will be in sync, compare transferred zone data with the master's zone data using AXFR.

Zone transfer test – This test tool send JP SOA queries every second to JP's zone transfer server and target DN server. And (inaudible) trace between transfer and the target, this (inaudible). And after zone is in sync, try dig JP AXFR to JP's zone server and test target and compare.

Some results of zone transfer test – If the DNS server is located overseas, AXFR transfer may take large time – sometimes takes over 15 minutes. If the DNS server's connectivity is poor, the test tool sometimes cannot detect SOA changes. On my test I found old Bind 9 stops responding queries while it is dumping zone backup file immediately after AXFR – because dumping of JP zone takes five seconds, and my tool detected five seconds' no response; it is fixed in Bind 9.7.1.

Response performance test – using queryperf and we did two test cases – no_error case and name_error case; to the target DNS server.

DNS response test – the goal is that the software answers all queries correctly; setup both the test target and the reference as JP slave; send all possible queries to both reference DNS server and target DNS server; and compare all responses.

DNS response test – both target DS server and the reference server are JP slave DS server and test to send all possible queries to both reference DNS server and target DNS server and compare to all responses.

It is possible queries in JP - owner names from JP zone. Registered domain names and glue host names; non-existing name; empty non-terminals.

And 28 patterns of domain name and query type – domain (inaudible) and so on. And non-existence domain names for maybe non-existence name and type A, AAAA, MX, NS and so on. And three attributes – noEDNSO, EDNSO and DO-1.

Total queries – JP zone has about 1.3 owner names including glues; times 28 patterns; times 3 attributes; times 2 servers makes 218,000,000 queries. And my test tool sent the queries specified time steps – 1 millisecond step case, it sends 500 queries per second for both servers; the test takes 218,000 seconds, about 3 days.

The comparison on DNS response test – there are different DNS responses but they are correct DNS responses – ordering in the sections; additional section may contain glue RRs; authority section may contain zone's NS RRs; EDNSO payload size may be different.

Correct differences need to be treated as no problem – if I find a difference, I evaluate it is okay or not; if okay, I need to update the comparison program, not to report the difference; and I don't know how to automate the step.

Some findings of DNS response test – When I found some bugs, I reported and they were fixed or I didn't use the software. The Bind 8 was old – it put NS RRs in answer section at delegation; recent DNS servers put NS RRs in authority section. Bind 9 sometimes changed the response patterns – recent Bind 9 does not add authority section in DS or DNSKEY answer to minimize DNS packets. And it is a sum-defined comparison.

DNSSEC and Non-DNSSEC response test – I prepared test signed JP zone and load it into the test target – added some DS RRs and signed. Prepare reference DNS server with traditional non-DNSSEC JP zone; I sent all query patterns to both servers and compared responses. Ignored differences of DNSKEY, RRSIG, DS and NSEC3. This test resulted that resolvers are not affected by JP zone signing if the resolvers doesn't perform DNSSEC validation.

Conclusion – trying all of possible query patterns are very useful for DNS server evaluation, even if it takes very long time. There were many bugs. We are trying to avoid bugs on our DNS servers. DNS software/service evaluation is important for JPRS.

We would like to know – Do you evaluate DNS server software/services on a user's point of view? Comments and questions. That's all.

Eberhard Lisse: Any questions? One here.

Peter Cox: I'm Peter Cox, [DN-NIC]. What you've presented is more or less a full regression test between a new version and an old version, right? So if I understand correctly, you're using the whole JP zone during this test. Have you thought about just picking random representative samples to speed up your tests?

Because I could imagine that you want to run the test more than once when you change configuration values because that's a problem that we've been facing – to answer your last question – that a new version comes out, a new version of Bind, say, and you have a plethora of new options and features and so on and so forth. And you want to find out what's happening. But, of course, testing the whole zone is a bit cumbersome. So what about representative samples there?

Kazunori Fujiwara: Testing takes very long time but some (inaudible) testing, it's useful I think but a full test takes only one or three days, so the truth is maybe possible. So we (inaudible) JPSKEY for the test.

Peter Cox: Okay, one follow-up. So not to blame anyone, but sometimes you have to deploy a new version and don't have three days available for particular reasons we're not going into right now. So I did say I do not want to blame anyone.

But still you want to do due diligence and test the software beforehand, so that's one of the opportunities where we'd like to be able to have a speed up in tests, but we're definitely looking into this. Thank you.

Warren Kumari: Warren Kumari, Google. You said that one of the reasons that your life is harder is cause the JPRS zone has everything in one zone or the JP has everything in one zone. You said that your zone is more complex because you have, or more complex than other TLDs cause you have everything in one zone. Why do you have everything in one zone?

Kazunori Fujiwara: Hmm… There is history on this. In the one zone case, (inaudible) signing is easy because the key measurement is only one.

Ondrej Sury: This is Ondrej Sury from cz.nic. Maybe there is a place for the NSOR to create a platform for them to do derivation tests on new versions while collaboratively because, for example, I know that [Nelinet] Labs has a Radisson suite for NSE. So maybe there is more room for a corporation in doing this GNSR platform for benefit of all of us. Right? It was more for Warren than for you, the comment. There may be work for the NSOR to run this sort of tests.

Warren Kumari: Yeah, it's one of the things we have considered. It's just a question of funding resources and time. But, yes, by all means.

Paul Hoffman: Paul Hoffman. To actually follow-up on Peter Cox's question, finding representative zones is actually extremely difficult and it is isomorphic to the question that he had said earlier which is how do you tell when a mixed response is actually the same as when it's been mixed up or not, trying to say this zone is representative of all of the edge cases we worry about. It's extremely difficult.

It's not impossible, but you inherently limit the edge cases and you end up doing a lot of research on what does an edge case mean to me and which edge cases do I care about. I think if you do that, you'll end up with a test that is extremely small, and you can probably run it in under 10 minutes. So that is a very different test

mechanism than what JPRS is doing, which is they are being purposefully exhaustive because they don't know what they consider to be an edge case.

Byrom Holland: We have a question remotely.

Sebastian Catsro: How do you compare the responses?

Kazunori Fujiwara: I translate the responses to a (inaudible) format using [R-NET] DNS and compare it during…

Eberhard Lisse: Thank you very much. Next up we have George Michaelson.

George Michaelson: So this is a talk that I gave earlier this year at the NSEC NOG meeting, so if you were at NSEC NOG, this is version 2. It's really a [BIS] talk. This is a talk I'm probably going to be giving at the IETF, so if you were at New Zealand, I'm really sorry and if you're going to IETF, I'm really sorry cause you're going to see this talk more than once.

It's really a talk about a IPV6 corner case. Yeah, that's a bit of a corner case. So the context here is that if you look at the

delegation of servers for the IP6 ARPA, you get a set of six labels. And if you look at the E label, you get an IP that happens to be in Asia, and if you do the reverse on the IP that happens to be in Asia, you get me. So I'm running one of the six listed NFs for the IP6 delegation space.

So the short version of this is that there is a lot of unbelievably stupid DNS out there, and V6 has invented a whole new class of it that if V6 succeeds, it's going to represent a really quite large traffic problem. So this is a proposal to augment the AS112 project and do a delegation of these queries before we have a real problem so that we can go out and get on with our lives.

The longer version of this problem is just to refresh people. If you do a look up for a [AAAA], you get an answer and if you do a look up for the reverse, you get a name, and that's all well and good, good and simple. But it's actually doing a lot more under the counter. There's this amazing profusion of dot labels; the strings are a lot longer than we thought; there's all these cut points. It's rife with mistakes.

There are actually people who are managing to mistake their reverse DNS delegation in 6. It's getting really very ugly. That isn't actually the real problem. The real problem is this thing about the cost of getting an active and the fact that there's a lot of negative answers that have to be given and that includes these V6

address types that everyone thought would never occur in global networks. But, unfortunately, they are being looked up.

This is just a graph from our DSC and in green, the color which means bad, you'll see there is a large spike over at the 800 point which is the increased size of negative replies when you have to add in DNSSEC payload. Previous versions of this data actually the two lines coincided around the very high spike at the 100 point, but as a result of DNSSEC, we've now got separation of the response sizes and giving a negative answer.

It's somewhere around three to four times more expansive. If this is a problem, the question would really be can we quantify the problem. I mean, what kind of stupid questions get asked would affect how many you can see and how expansive it is.

Okay, so the questions. Well, there's just so many stupid questions. The one down the bottom – undelegated – that's basically what AS112 is dealing with in IP v.4. It's handling things that no one ever expected to have global delegation the questions get asked.

The problem in v.6 is we invented all of these new types of address – different forms of scoping, the multicasting, the unique locally assigned and the tunnels. So to give you some numbers on this, this is what a typical day looks like on the server in question and

you'll see around the middle there that we're doing about 341 million queries a day looking up PTRs on this machine.

And if you look down the bottom, you'll see that the ratio of v. 6 to v.4 requests is about 7%. Now you've got to put this in the context that 5% of everything has to have a negative answer and that's a minimum 5. We actually see this going up as high as 40 or 50%, depending. So 5% have to have the negative answer, but at the moment, we're only doing 7% of the queries in v.6.

So it's not that we have a problem now, but if v.6 takes off, then the percentage that a v.6 won't be 7%; it'll be 20 or 30 or 40 or 50. And if v.6 has introduced all of these new kinds of stupid questions, then the risk of the amount of traffic that potentially starts flooding into the service is really high. And I stress its risk management. We're not really saying right now we've got a problem. I think this is about trying to preempt a future problem when v.6 takes off.

Okay, so just to drill down a bit into these stupid queries. A lot of people say we're never going to have ULA, the Unique Locally Assigned Address. If it's a bad idea, it's not going to happen. And I've got two slides here with people talking about the emerging 6 low pan technology using at the internet of things and what I've been told basically is it's very, very likely that these deployments of things like smart meters will use Unique Locally

Assigned because they're a really bad fit for classic global unicast assignment.

And the point about DNS is it's like the visibility of these things happening. It leaks information about these things leak. The packets never leak the local net, but the DNS queries do. So if we get 40 million people in California getting equipped with a Smart Meter and everyone of them is running on the ULA, and somehow this winds up being connected to the global internet – and you better believe it's going to because everything winds up being connected somewhere – we're going to have a lot of DNS queries.

So here's what I'm actually seeing right now per day. I see 1.4 million requests for reverses in the Unique Locally Assigned. And remember, this was never meant to exist; we didn't think this was going to happen. We're already seeing 1.4 million of them.

Okay, so the link locals and the site local. Now I'm going to break sideways to the live demo. I've got a packet catcher going on the Wi-Fi at the moment with a TCP dump and do you see scrolling up there all the FE 80s? Everybody in the room who has an iPhone, your phone is doing Bonjour, local rendezvous all the time. It's just doing it all the time.

Everyone with a Windows machine, it's doing this all the time. IP v.6, ICMP, it does promiscuous requests – "Hey, who are you? I'm here. Okay, that's cool. That's really fun. I've got local

roots. Want to have a local root? Yeah, why not have a local root. Yeah, who are you again. I've forgotten. You've got a long name. I can only remember seven digits."

So remember, I don't actually think any of you guys – because you are all true researchers and you are listening to me – none of you are doing anything on this network right now – but as I've been talking, this has just been flooding past – link local, link local, link local, link local, link local – and it only takes one box on each subnet to log this and log this with name to address look up and suddenly we're in a bad place.

So, nobody ever uses site and link local. Well, they do and it's MTP, it's the ARPA, it's the router detect, and I'm seeing between 2.5 and 3 million a day of this class of query. So v.6 is inherently chatty. There's a lot more traffic going on than people give credence to.

Okay, then we've got the whole question of the CorDECT A joint look up, the Happy Eyeballs thing, the thing that they find v.6 and it tries to connect. And then when it doesn't work it tries to link local and then it tries something else and then it generates a log.

So then we get to tunnels. Now this is really, really cool. Tunnels we all know are not really where we want to be and we've actually done a lot of work in the code to suppress them. So at the moment, at this position, if you look up a CorDECT in the DNS, and if you

get back an answer that says, "I can do this," you won't use your [terado]; it's suppressed. The code has been written to say, "Don't go there." But if you get a v.6 literal, if you bypass the DNS, it does go there.

Okay, that's no problem. Who would use the numbers raw? I mean, no one in their right mind would do that. Okay, so look at these numbers and notice that it's a log scale, a log scale. I'm seeing 10 million [terado] DNS requests a day and this is for a protocol that we reckon is not being used because it's suppressed in the DNS.

There's also around 10,000 [six-to-fours] so do sums on adding those together and you'll notice the [terado] alone swamps the global Unicode. So there are more DNS queries relating to attempts to use tunneling than there are real DNS about real v.6. That's really bad.

So we had to trick for 6 to 4. We did a delegation and we have an engine that allows people to register; a [terado] because it's an ad hoc tunnel mechanism that is done per session binding. It's a lot harder to do this. We thought really long and hard about saying, "That's okay. Let's delegate this." It turns out it just has really bad scaling properties.

Okay, mapped addresses. This is the royal telephone from the summer palace in Beijing and you'll notice someone has retro-

fitted a keypad onto it. So this is the case where you think you have 6 but you don't know a prefix so you just whack it into a v.6 address with v.4 and bang some zeroes in there and let's see what happens.

This was never meant to get off the local net. Four million – okay, look, it's good; it's good. The numbers are going down. We should be so proud. We've suppressed this problem. It's only 4 million. Okay, so the end point of this is that again on the log scale, we're basically doing two decimal orders of magnitude more silly queries than we're doing purposeful queries in reverse. And this is with 7% query rate on v.6 because the functional uptake right now is kind of around 1%.

So imagine that the future comes true – that we get what we want and we get v.6. This problem can only get worse. So if you're not fully up with AS 1.1.2, this is some work; Joe Apley owns a lot of the documentations data on this and there are two drafts that you should be reading. The first one is the client's eye view explaining why you're going to get spurious packets back from AS 1.1.2 and don't worry about them and the bottom one is the operational context.

And what I'm proposing is a request to the IAB that basically asks for a delegation expansion to include in AS 1.1.2 the labels which would help terminate these bad address cases.

It really is a one paragraph draft but you have to put in about five pages and get out some clause and three clause Berkeley license and God knows what in there. That just hit zero draft statement is going to get discussion in DNS Ops and Peter will be terminally bored because he's going to see this slide pack twice, although I think I probably have to put more technical content in. Yeah, he's nodding.

But there's some stuff you really do need to do. On referencing the Bind confix because I just don't know what other name servers do. But if you get a newer spec, you actually get local delegation of these zones. So there is software out there that tries to help minimize and keep the zones local. But even bearing that in mind, I think we're probably going to need the delegation change.

And I very strongly suggest that people look in their logs because you may well think a bunch of stuff you're doing is private, and I can assure you, information about it is leaking out into the global space. So I don't want this to be like I'm saying, "Oh, no, v.6 won't work." This pack only comes true if v.6 works. So this story is about a problem we get if v.6 takes off. It's not an attempt to kill 6; it's an observation that 6 is going to kill the DNS. Okay, that's me.

Eberhard Lisse: Questions?

George Michaelson: We can finish early. That's good.

Eberhard Lisse: Yeah, no questions? Oh, Peter has one.

George Michaelson: Oh, goddamn it, Peter. [laughing]

Peter Cox: (inaudible)

George Michaelson: This is my collection of thick wire Ethernet. I'm determined to get it back working again.

Eric Ziegast: Hi there. Eric Ziegast, IFC. Question – I'm one of the…

George Michaelson: You're one of the good guys.

Eric Ziegast: We will operate one of the nodes in an [any-cas] fashion. Are you planning on doing the same thing?

George Michaelson: I am asking for you guys to be given delegation of these labels. So if you are one of these nodes, you're going to get an operational instruction to include responses for these zones in your current service.

Peter Denning: Peter Denning. First I have to apologize to George for nodding at the wrong point in time. I'd like to reflect on what Roy said a couple of hours ago about solving the problem at the wrong place. Because you're actually stepping ahead here, going to deploy an infrastructure that is covering other peoples' mistakes.

AS 1.1.2 has a sibling which is suppressing these queries on the resolver side, and some of the protocol specifications that these fancy new technologies actually are based on actually explicitly say thou shalt not issue these queries in the wild.

George Michaelson: Nobody reads that part of the spec, Peter.

Peter Denning: Yeah, sure. I'm so sad. The interesting part and that's the question here. You've come up with an interesting data point about where these questions originate. Like people are doing bad things, like you letting run TCP dump in Map Address…

George Michaelson: I did TCP dump minus m. I suppressed my DNS.

Peter Denning: Yeah, you did.

George Michaelson: Hang on a minute. I should turn that off. [laughing] Let's not do that anymore.

Peter Denning: You did; I didn't. The real point is here – do we have enough of an idea how these – even if some of these infrastructure parts would behave benignly, how widespread the problem is; how many codes would have to be changed?

George Michaelson: Yeah, these are really reasonable questions and you're right. I leapt ahead because I'm trying to get my name in lights in publication. Hey, we all know that's what the IETF is for. No, but seriously, there probably is a better way to deal with this.

And we put the draft out maybe in ahead of saying there's a discussion of problem space and led to a solution. But really, the draft should be asserting there's a problem and I think you're right to say, "Let's understand the problem; let's categorize it; let's look at different ways of solving it because this may not be the solution." Maybe the solution is to get to the code points and say,

"Don't do that." Could be true. Hey, just means a version 2 of the draft.

Eric Ziegast: Eric again. One of the things that we did with AS 1.12 with IPv4 is make the data available so you can see. There's one argument that says, "Well, we should let it fail because it shouldn't be happening and therefore things might actually get corrected."

Another would be, "Well, you can actually find out who are the people who are sending these queries and go ask them," and that data is available at SIE and we make it available to OARC as well.

George Michaelson: Fujiwara san, in his slide pack, mentioned the PC's a population of around 1.8 million IPs that are the consistent set of IP ranges that perform queries on him. That number really interests me a lot because I've done similar exercises looking at the half-life of IP addresses querying into my infrastructure. And the number I see is tending towards the same kind of number.

So if we said for just sake of argument, it's a population of 1.8 million addresses that are doing DNS resolving, my gut feel is that they distribute reasonably evenly with the /24s and the /16s that are the root announcers of the global internet.

This problem is essentially fully globally distributed. You could certainly phenotype it by fingerprinting. You can get into the packets and say, "I can tell this is a query coming out of a Microsoft machine."

You could get into the other behaviors and start to say, even for the v.6 queries, "I can tell from the Mac address what the platform is that's probably doing this," cause I've done that and I can tell certain classes of query that have to be coming from Apple's OS because it's just Apple Macs that do them.

So it's widespread, but you're right to say we could do work on the logs who's doing it. And I think exposing that and getting some work done would be a mighty fine thing to do.

I'm interested by the numbers though - 1.8 million that go to places in Japan. I reckon it's about 1.8 million that come to me. I wonder how many of them are the same 1.8 million IPs. Maybe that's our target audience here. Maybe we just have to go out and kill 1.8 million resolvers and we have no problems anymore. Oh, hang on. Wrong movie.

Eberhard Lisse: Alright, thank you very much. Next up we have Paul Vixie. Do you want to use your laptop or do you want me to slave off mine?

Paul Vixie: Hello. So, I'm Paul Vixie, speaking here as ISC. I want to begin by saying that when K.C. Claffy and I first put together the idea for a DNS OARC, most people thought that we were crazy and had a very hard time understanding why the industry needed something like this or what it would even be like.

Now that I have been out of touch, off the Board, uninvolved for some years, and I see that it has grown to this, I want to say this is wonderful. And thank you to K.C - I think K.C. is listening – for getting this organized with me.

So I'm here to talk about response policy zones – a controversial topic. A number of people think that it is the next instance of pure evil on the internet. I will probably justify those claims and then refute those claims all in the next 15 minutes. Next slide, please.

So DNS works really well. It's a large system and one thing that's true about large systems is that there's always something broken somewhere and yet DNS works really well. A lot of servers, a lot of operators, lot of different implementations and yet, most of the time, people get what they need from DNS. The failure rate is crazy high, but the success rate is higher.

The problem from my point of view is that it works as well for evildoers who want to use the internet to execute internet crime as it does for sort of the rest of us who either want to prevent that crime or just want to do our business. So I'm interested in creating

differentiated services. I would like the internet to work less well for bad guys and that's what this is about. Next slide.

So this thing about the decentralized systems distributed autonomous reliable hierarchal database – it's great and DNS is the first thing that is all five of those things. There may be other databases that have been created since then that also have some or all of those properties but DNS was first and it's definitely the biggest.

One of the ways that it gets its resiliency is by having a fairly loose tenuous relationship between the different players. You don't need strong contracts or some kind of a leased line between you and the next guy up or down the DNS chain. All you need to do is register something somewhere and everybody can reach you. And that's the reason for the success.

If we required strong contracts or physical co-location or money changing hands with everybody who does anything with DNS, it probably couldn't have gotten this big. Part of that resiliency changed quite a bit during the international forum for the white paper which became a green paper back in the early days of commercialization privatization of the internet.

The .com was then called a monopoly, but the .com domain was held by a single company who was making a lot of money and standing to make quite a bit more money from registering quite a

few domains. And a number of folks said, "We want our share," and the technical people said, "Well, there really isn't a way to share it because there can only be one set of name servers," and so forth.

And so various people came up with this model which splits the registry and registrar functions. So the person that the registrant has a contract with is the registrar and the registrar then has contracts with the registry and then revenue gets shared all along and people are generally happy.

So the problem with this is that there is no accountability. If somebody reaches out and touches you in a way you don't like on the internet, and you know what their domain name is, chances of you being able to find out who they are so that you can either prosecute or sue them are low. Anybody can do pretty much anything and they can get as many domain names as they want, usually at a very low cost, sometimes zero cost, depending on what loopholes they know.

What I'm getting at here is that the split registry/registrar model has decreased overall accountability and I know that there are working groups about WHOIS, I know that the law enforcement community has got similar complaints and our working groups in law enforcement also trying to improve accountability, maybe get us back to where we were before. Where we were before wasn't good enough either, frankly.

So in the absence of accountability, we're going to have to find some other way to manage this differentiated service level that I would like to create where good guys get better service than bad guys. Thank you.

Now, DNS was not the first unaccountable system on the internet. Those of you who receive email are probably receiving a lot of spam. You may be paying a lot of money to various companies that are helping filtering it for you. I suppose it was good that we had all that spam because we've created quite an industry for filtering it. But the fact is anybody can send anybody email.

There's no accountability there either. And the people who want to send you email that is an ad for something you would never think that you would ever want to buy is unbounded cause their costs are zero.

Your costs in handling it are not zero. That's an asymmetric cost benefit relationship; you are incentivized to make it stop. The ISPs who are allowing it to touch you are not incentivized to make it stop, therefore, it grows, it does not stop.

We created – and I say we because Eric is in the room – we created a system years ago called the Real-Time Black Hole List and it started out as a BGP feed and then Eric Ziegast – stand up; there he is – that's the guy – came up with a schema that put the Real-Time

Black Hole List into DNS so that we could then make lists of people who were sending a lot of spam and then reject all that traffic from them.

Some of you know that at the end of that story I got sued out of business and I no longer do any kind of reputation services. But if you're sort of in the security field, you know that there are hundreds now of commercial services that are based on the schema that Eric created for that – that Real-Time Black Hole List. And that was all because of the unaccountability problem.

We need to do the same thing now for DNS content as we did, whatever, 15 years ago for SMTP source addresses. And that's what we've done with the Response Policy Zones. We released version 1 about a year ago, thanks to Barry for coming into ISC and saying, "What is this and why has it not been published?" So it went out.

In the course of piloting it we discovered that a bunch of things were incomplete. It was wrong for what people actually wanted in this space so we have made some improvements. But first let me explain that in version 1 we had a very simple rule-based system that can only be triggered based on the query name and the query type. And based on that trigger, you would get back some kind of a policy override that would, say, return some kind of fake answer - either a fake CName, a fake NXDomain, a fake positive answer.

It was a subscription model and it still is; that has not changed. We don't intend to ever send a wide-area query to some distant DNS server to learn how to respond to some other query. That would be nuts. We would at least double the amount of traffic on the wide area if we did that. So we're not doing that. You have to subscribe a recursive name server to the policy source in order to have this effective on your real query stream.

And all of these rules were encoded as a zone. So we overloaded and misused and reused a bunch of different RR types in order to express policy inside of a zone. And this is because I've learned that if you want to subscribe to something and you can do so using the same protocol that the firewalls are already permitting, then you can deploy faster. Otherwise, you're counting on a whole bunch of people updating their firewalls before other people can then subscribe to your content. So it's a zone.

And what we've learned by piloting this is that it's not good to only be able to trigger based on query name and query type. Sometimes if you know that something is bad, and you would like to put some policy around how you're responding to those bad things, the thing you know might not be the name.

What you might know is what address it's going to return. That could be the anchor point to your policy. You're sure that a given CIDR block has got nothing but bad in it or you're sure that a

certain IP address is currently part of a botnet or it's a botnet command control server or something.

What you're not sure of is what name they're going to be using for the next 20 minutes. Maybe they're rotating names very quickly. The other thing you might know is you might only know the name server name. It's possible that all zones served by a certain name server are gonna be full of nothing but bad. But you won't know what zones those are in advance. And if you want to have policy that is effective against that particular malicious stream, you need to be able to trigger your policy based on name server names.

These were things that I suppose I should have thought of before we launched 1.0 but I was glad we were only in sort of a pilot Beta phase because we fixed all of this in version 2. And it's in Bind 9.8.0 at the time these slides were prepared, 9.8 was in Beta, but it is now in full production.

I know a lot of you don't like to run .O releases on your production systems. I suppose at IFC that means we should stop treating them as Betas or we should stop making a separate Beta release for .O releases. We should treat it as a Beta, but we don't.

Anyway, it'll also be in 9.8.1 whenever it comes out and I'm hoping that those of you who are interested in this will start putting 9.8.0 into your test labs now. I run it in production at my house; I am a power user and it hasn't hurt me. But, of course, those of you

with millions of customers may want to do more testing than taking my word for it. Next slide.

So, I'll give you a couple of short examples of what these zones look like internally and I apologize for the low quality of the graphics on these slides. A CName to dot cannot occur in nature. If you do it, it's pretty well meaningless or useless or even harmful. So we used that as a trigger to say, "Gee, if you see something like this, you ought to return an NX Domain instead."

So in the first bullet here, if somebody asks any question about example.com and the recursive servers subscribe to a policy zone containing what's in this bullet, then they're going to get an NX Domain.

You have to separately handle the sub-domains. So you can, in this case, if you only had that first bullet, you'd be blacking out the example.com, but not the sub-domains. That's a sore point with me because in my opinion if that domain doesn't exist, no sub-domains can exist either, but other people don't agree. That's not exactly what the standard says.

So, anyway, we have separate capability for forcing NX Domain for sub-domains and that's where you put the asterisk dot in front of it. You can also substitute a positive answer as is true in the third bullet here so that if that is the question then this will be the answer, no matter what the truth might be. Next slide.

On this slide we see that we're – I'm not going to get into the protection thing. Forget the first bullet; we don't have time for that. You can force an empty answer; you can force a certain name to be empty node, empty terminal, empty non-terminal, whatever.

The third bullet is where it gets interesting for version 2 because that's how we are encoding a CIDR block where, if the answer is in that /24, then it will be substituted with the answer that's given here. You can, of course, also say that if the answer would be in that /24, you'd like to give back an NX Domain for it. So you can have walled-garden behavior, you can have just it's not their behavior – whatever you want. Next slide.

Came up with kind of an interesting encoding for doing this in v.6. The colon-colon syntax that we use in v.6 addresses, v.6 literals to represent the longest string of zero bits in an address, it turns out colon-colon is not illegal in DNS, but Bind happens to complain about owner names that contain dot.coloncolon. in them and so rather than fight with Bind, I just decided we'd use zz to replace the colon-colon.

So in this example, we have a /48 and we didn't have to put .0.0.0.0, etc. Fixing Bind was much easier when ISC only had one person working for it and it was me. Now we have customers, we have people who say, "Wait, if we change that, we have to document it." Okay.

So here's what the config file looks like. I'm sorry the font is a little small. In this example I've subscribed myself to six different policy zones. In the first one and the second one, I'm allowing the policy to be expressed in the zone and that was all you could do in RPZ v.1. But in RPZ v.2, you can also say, "Look, I want to get the triggers from these policy zones, but I don't want the publisher of the policy zone to be telling me what walled-garden to use or whether to use NX Domain or whatever."

So in the last four examples there – policies 3, 4, 5 and 6 – we're only looking at it for triggers and whatever the policy that is expressed in the policy zone is ignored and it's overridden and you'll get what is shown there.

And the zones themselves are normal slave zones. In this example we didn't use TSig because that would not have fit on a slide, but, of course, I'm expecting people to use TSig and also expecting them to say "allow query – none" because it's probably not good if you let other people find out what your policy is because then they could bypass it.

So on the producer side, I've done this a couple of different ways now just to sort of find out what kind of trouble you're in if you're a security company and you have an existing database and feed of malicious trigger information and you want to publish it in this format.

I've done it by using RFC 2136 updates – dynamic updates – that works fine. Bind will aggregate those and will periodically IXFR to its slave servers and will do the notifies and so forth.

But you can also just have a chron job that extracts from a database into the zone file itself once a minute or however often you want, and tell Bind go ahead and reload that if you turn on this option called IXFR from differences. In that case, Bind will load the new zone, compare it in memory against the old zone, generate the IXFR log that is necessary and then do the notifies and then it will send only the deltas down to the slaves.

I have done this with domains' policy zones that – artificial, of course – but policy zones containing 3 million different names – actually, 3 million rules which, because of the wild carding problem becomes 6 million names – and then I have changed it by 10,000 per minute and it takes about two seconds of CPU time on a computer about the same performance as most of the laptops I see here. So we're really not talking about a big problem.

So it turns out that using zones for this – although my choice of zones for this as the bearer channel for this – was because of firewalls – it turns out that they have exactly the properties that you want – it's efficient; it's protected; it's authentic; it's minimalistic – it really does, it works well. Next slide.

So there's good that comes of this. Specialization of labor is how economies of scale are built. We can't have it be that every potential victim or even every ISP of every potential victim, does their own security research to find out where the bad guys are and what names and addresses they're using. Otherwise, the bad guys will have us overwhelmed, kind of like they do.

What we need is a smaller number of companies and people and analysts that are in the business of finding out what the bad guys are doing and offering the results of their research both in fixed time and moving time research to people who would like to maybe reject all traffic that involves those triggers.

You can't do that unless there is a common framework where the people that know this stuff and the people that want to know this stuff can communicate. Now Bind is not the first thing to implement this and RPZ is not the first format that's been used. We are, however, the first open format. There's nothing licensed about this; it's all freely available. I encourage other DNS recursive name server implementers to implement this; I encourage security companies to go ahead and start offering their feeds in this form.

To the best of my knowledge – I haven't really paid lawyers to do a search – but to the best of my knowledge this violates no patents. It's something that we could all start doing. And the ultimate impact is, I think the best case right now, if somebody registers a

dot org or dot com or whatever name, and they're using it as part of a phishing attack where they're sending a lot of spam and it's something or other that's got PayPal on the string and it ends in dot org, and you want to get that taken down, best case is a couple hours; average case is never; fairly often it's a day.

With this, in my testing, it's about five seconds. We can do something with that because the bad guys can execute a lot of crime inside the existing time window of two hours or a day or never. They can't get a damn thing done if it's five seconds. So we can push them out of DNS and make them use something else, which means we'll have to track them there next, but that's about the best we can do since locking them up seems to be out of the question. Next slide.

There's possible harm that comes of this. This is a controversial topic. I've had a number of different employees of different government organizations call me on the phone and say, "Look, we have to do [COECA]. We have to force ISPs in our country to protect the Hollywood entertainment industry against pirated content. And I saw this RPZ thing. I'm wondering if I should make that a recommendation so that we can have that be the force of law."

And what I tell them, all of them, is that the moment that Facebook.com is hosting some pirated content, you're going to have the choice of turning off the information economy or letting

them slide cause domain names are a very blunt instrument. And a given domain name might have a lot of good and a lot of bad content on it at the same time. So no, you really don't want to use this for that, but then I have to admit it would work, technically speaking, this would work.

It's not what I designed it for and it's a bad tool for the job, but in fact, it would do what you want. So I hate that idea. I hate the idea that this is likely to be used in ways that are stupid. But that's inevitable. All tools will be used in ways that the maker of the tool thinks they're stupid. If I told you some of the things I've done with hammers, for example.

So there will be a number of RPZ feeds that are either racist or secular in nature. I can imagine some vibrantly Christian community deciding that they want their customers to not be able to see any Muslim websites or vice versa, for the Muslim community and Christian websites. I think this is stupid, but I know that people will do it and they're going to use my tools to do it and that bothers me some.

This will make DNS less reliable. Those of you who have help desks or have knocks and periodically get trouble tickets like, "Gee, I tried to go to www.example.com and it didn't work," you're already looking at the timestamp on the trouble report to find out if maybe the name server that serves example.com was

somehow removed from BGP at that moment and that was the reason and so that you can close the ticket.

This is going to add a lot more things that you will have to check to find out – did the reputation system cause this problem. Or if so, which reputation system because you can subscribe to more than one. Your customer may be subscribed to one that they're not even aware of. So I am going to make everybody's life harder and let me apologize in advance for that. But I think there is more good than harm. So in spite of the harm, I am pushing this forward. Next slide.

So, that's my email address. In case you aren't one of the millions of people using it already please contact me if you have further questions. There is a mailing list where you can join and discuss this. There is an online version of Specification. It looks an awful lot like an internet draft, but it isn't one. It has not been submitted to the IATF because that is sort of 10 years of pain that I'd like to postpone as long as possible.

And I wrote a blog post about it. You can Google for that. Tried to put the URL on the slide but it wouldn't fit, but this fits. And, of course, I'm here so questions, comments are welcome.

Jay Dailey: What can't you configure – Sorry, Jay Dailey. Why can't you consider it to return a servfail?

Paul Vixie: Because we didn't think of it and nobody has asked. Can you describe how that could be useful, how that could be helpful cause creating new sources of failure was really outside the box thinking for me.

Jay Dailey: I'll have to refer to my technical next to me, but I think it's possible through doing that to have less bad impacts on DNSSECs with the use of this.

Paul Vixie: Thank you. That's an idea for RPZ v.3. Every time we rev the format, we make sure that old policy zones still mean what they always meant. We just add some new meanings that will be ignored by older implementations, so that's a good idea. We'll think of something. Perhaps we'll cause it to be a CName to a Jay Dailey. org or something as the signal for that.

George Michaelson: Paul, have you really invented yet another representation of addresses in reverse under a label when you took the hex nibble blocks on the colon boundary and embed them frontwards in a reverse sequence. You've invented Intel byte order in the reverse name map.

Paul Vixie: If I had them frontwards on the slide – can you go to that slide, please – then the slide is wrong. No, it's backward. Although it's frontward per nibble, so yes, it's Intel Floating Point. So the point is – forget the point for a minute. What we were trying to do was to make the zone compact and we very much had to make the assumption that on the implementation side, in the recursive name server, whenever you got a new version of the zone, you were going to update some set of data structures that was not the zone itself that was in parallel to the zone, and it's probably a Patrycja tree.

And so we didn't care so much what the format was so long as it was short. And we just had to make it short and then we parsed all of this stuff as we received the rules. Any other questions?

Roy Arends: Roy Arends of Nominet. I've got two technical questions really. I assume that the 9.8.0 has no default setting on this. There's no default group policy zone. That's correct? Okay, that's good. The other question is when you add CName to a root label, what happens on the client side? Does it do an additional query and to the root, or is Bind clever enough to not just follow anything then?

Paul Vixie: So I want to emphasize we are using resource records in a new way here. They are expressing the policy that the recursive name

server ought to follow. We never, ever follow any of these CNames. We will return some of them. The format that is used to indicate that a walled-garden should be used – that CName gets returned.

CName dot is never returned; it's an internal signal to say, "Just answer with NX Domain." So have no fear; I was not thinking as a root name server operator that I needed even more junk queries.

Eberhard Lisse: Anything else?

Sam Wyler: Sam Wyler. What happens if you have an intermediate resolver that doesn't [groc] this? Does it follow the bad CNames?

Paul Vixie: This data, I suppose if you don't restrict queries to these zones and somebody does look them up, they will get the CName and follow it; that is true. But the names themselves are expected to be a lot like TSig names where they aren't in DNS at all. There's no NS records in these zones unless it's just NS for local host and not delegated anywhere.

You can't do a wide area query because even if you knew who the servers were, they would be doing a live query none. So it's very much an alternate name space type of thing where, if the names

that you're using as anchors for just the zone names – you have to use something – happen to be the same as whatever your company uses in the real internet, that'll be just kind of a coincidence. I see no more questions. Thank you very much.

Eberhard Lisse: Thank you, Paul. We have Eric Ziegast up next.

Eric Ziegast: Hi there. My name is Eric Ziegast. I work with ISC. I work with a bunch of – I actually see a whole bunch of them here this time – we have some operational types and we have… I do a little bit more on the security side. The next slide.

ISC does a whole bunch of things. They write software; they host public benefit stuff; they're starting to do a lot more security things. RPZ is just one of them. Security information exchange is my bailiwick. I help shovel a lot of data between the people who have it and the people who need to use it as far as going after the bad guys. Next slide.

DNS amplification – what is it? It is a method of attack that's commonly used, I think most everyone in this room is familiar with it. Hopefully most of you haven't seen it, but it's out there and it is basically used to send a lot of data down toward victim servers in a way that's just not accountable. It's very hard to trace, hard to go back.

There are a lot of contributing factors to its success. Some of those ingredients is that first you need someone to say, "Hey, I want to do harm to that network," or to that website or whatever it is. There are also contributing factors which is lack of filtering on internet service provider networks, such that it is possible to inject UDP packets into the network and DNS is a UDP-based protocol for the most part with some TCP fallback that will allow them to spoof an address for someone else. And there's not enough filtering on the internet to prevent those packets from coming in.

Another issue that we have out there is that the botnets or the bad guy operators are using open recursive name servers which are very, very prevalent on the internet. Just like with no insight you could really do really bad things with email – with SMTP. A lot of people had these ideas that, "Yeah, anyone can query my name server. What wrong could happen from that?" By default a lot of people just don't think about it. Next slide.

So generally here's a recipe for how a DNS amplification works. You start with a server that you can control. It's just a case where you're actually using a hosting service to go ahead and create the attack. You go out and you send an ANY query or an NS query or any kind of query that results in an answer.

Your query is only 36 bytes or something around that size, but the answer that comes back might contain a lot of data. And what it is

is that is the amplification. And what happens is that you might start out with a rate sending to one server, you may pick an open recursive server somewhere on the internet.

You say, "Alright, I'll go and send 10 packets per second, 10 of the same queries per second, it's going to open a recursive name server and oh, by the way, I'm going to forge the return address so that those answers go to some other server on the internet that I don't like."

Your bit rate is only 2880 bits per second and your victim is going to be seeing the answers come from the recursive name server, same number of packets per second but now you're going to get 144 kilobits per second, using if, for example, you had 50 times more data in the answer than the query.

Now, let's say that you had some really good scripting or some tools and you can say, "Alright, I'm going to randomly pick 3,000 open recursive name servers on the internet and now I'm sending 60,000 packets per second and I'm using almost all my bandwidth there, but, hey, it's unlimited. I only paid $50 a month and I'm going to use it. And I'm getting my 8.6 megabits going out. ISP might notice but do they really care?"

A little bit, but not enough to actually give you a call in the middle of the night to say, "What's going on." The victim rate, unfortunately though, is the same amount of packets per second,

but now they're getting the amplification and they're getting 432 megabits according to this math for this example, and the victim will surely notice because they usually have 100 megabit or a gigabit on their server.

And if you'd like to add some pain, add more servers just like this and you can get gigabits of traffic going to a particular destination. Next slide. And it's very hard to trace. The spoofing, all you see at the name server that you're using for the queries is you see the queries coming in from recursive servers. At the recursive servers, all you're seeing is packets that are coming from the victim, not the person who is doing it, and your packets are going out to the victim. But you have no idea where those packets came from.

Not only that, but a lot of people, they just don't have the tools that they need to actually know that this is going on and they don't have any idea how to figure out how to mitigate it. Usually what happens is, well, if you're at the target you say, "Oh my gosh, I'm having 2, 5, 16 gigabits of traffic come in at me. Help! Help!"

If you're a cursive name server, you're seeing it get abused, but perhaps your name server didn't keep up so you're not necessarily going to take the name server down. So even if you turned off the destination customer or the person who's the victim, that doesn't really change anything; now the ISP is on the hook for those packets and they have to deal with it. And eventually the attack

subsides because the damage has been done and say, “Alright, bad guys, we win.”

So next slide. In 2009 I went to a conference, we had a little workshop, we had a bunch of guys around the table and we said, “Alright, we’re seeing all these ./ns records coming in and we didn’t know where they were coming from; we were trying to back-trace it.” We weren’t successful that day. We run into the same problems that we have today and they haven’t been fixed.

But the thing back then was there are 13 root name servers, so your single query which is very small, is getting back something much larger. And in fact, they did things like you had A records and at some point [AAAA] records. It was really the method of choice for doing d doses. So there’s a really great write up; people at Secure Works, they’re really good at this. And in there there are actually other links to other articles. I recommend that you go look at the link at some point.

But in 2001, here we’re seeing on the network ISC.org/any and we get these things into our knock and show our support and say, “Hey, what’s going on? What is your name server doing to us?” Or, “Why is this name server on our network? What are you doing to my network? What are you doing to my name servers?” So we get that.

For some unknown reason people are going ahead and using our name for that. And part of that is because we're an implementer of DNSSEC. We have very large answers when you query in any record out of someone's cache. You can go ahead and try it.
In this example its return IS about 3,400 bytes or essentially a 95 to 1 multiplier. That works really good.

So whose fault is it? Well, it's obviously our fault. You should block ISC.org, right? I mean, obviously we should stop doing answers. No, that's not the answer. It must be DNSSEC. Obviously if there's a 95 to 1 answer, DNSSEC really sucks; it's bad. Don't use it. But I think the train has left the station there. So DNS amplification is not limited to ISC.org or DNSSEC. So it's a general problem that has to be dealt with. Next.

Perhaps the reason why we're being used for this is that there is some really good documentation on how to do it, as well as a tool kit. And you can go ahead and do it and go ahead and use it yourself and thanks to the developers of that. There's some rebuttal to why it is ISC.org or why it's a problem in general has to be dealt with. Gave you a little link there.

Not only that but they're gonna say, "Oh, no, no, no, don't hurt ISC. They're good guys, right?" Well, hackers love us and bad guys love us. We do things that help make their lives a little bit miserable, or at least parts of us do.

So in short, it sucks to be ISC in this case. Now it's not causing us too much damage. It's only a small percentage of our normal traffic in general that we see. But, you know what? It might not suck because now we're in a position where we actually get to affect what's happening with this. If this were someplace else that was unaccountable, we'd say, "It doesn't affect me," but we actually care so we're going to go look into it. Next slide.

The real problems – someone wants to inflict harm, what I stated at the beginning. There are all sorts of different ways that you can cause problems in that you don't have accountability for who you sell your servers to. Some ISPs try to be as lazy as possible or as cost minimalistic as possible to eke out their small profit and the cost gets shifted to the ISPs who are trying to do things right.

There's malware out there; there's botnets that are out there; there's lots of guns at the ready to go ahead and shoot down and do a DNS amplification tuck at any time. The bullets out there are the open recursive servers and the lack of filtering that's out there. And that's what is actually doing the harm. And the accountability because you can't trace it back or it's very hard, the crooks don't get caught so they get to keep doing it and doing it and doing it and it'll keep getting done until something happens. Next.

So BCP 38 and its best common practices for internet service providers and this is a diagram that actually came from the RFC itself. There's an attacker out there that wants to spoof some

traffic to make it look like it's coming from the 12 network. And what they can do is, because there's no filtering, and you can see my source address is 12.0.0.1.

By the time it actually gets to somewhere else - like their destination was the 11 network – there is no filtering there and this network, the 11 network, can't tell if it really came from the attacker or whether it came from the real network.

What you're supposed to be doing is filtering at the edge so that the only packets that are allowed to come at the router into the internet are the ones that have source addresses that match what the customer is supposed to have. If everybody did that – I don't think we'll ever get to the point where we can do that across the whole internet. Until most of the internet does that, there's not going to be much accountability for EDP. So we can try to do the best we can but we have to keep moving on.

So for peering, when you have lots of networks talk to each other, the typical guidelines for what you're supposed to use, just say, "Are you my peer? Should I be exchanging on a free basis or on a reduced cost basis between me and you?" They look at things like, well, do you have a 24-hour knock, are packet ratios about even, are we go ahead and you're just not dumping your data onto my network, are we actually in the same multiple regions.

That's the normal stuff but how much do ISPs really care about BCPs 38? If they actually were peers, one of them is a BCP 38 compliant network, they would require this from the people that they peer with because if they don't it's going to be a much larger security headache. The one that's not doing the filtering is actually shifting some of their cost onto the BCP 38 compliant network.

The compliant network has to filter traffic if they're going to insure their compliance talking to other BCP 38 compliant peers and there's the problem where if you have two good networks – A and B – and you have a bad network, C, A is going to be affected almost as bad as network C. Next one.

And it might be possible to verify the reputation of whether networks are doing BCP 38 enforcement or not. And you could basically say, "Hey, is there someplace where I can verify it?" If I spoof an address, then it's not going to get to my network. There are a bunch of different projects out there where you can go ahead and do those kinds of queries like a Looking Glass. Next slide.

Another issue is the open resolvers. Someone gave a talk recently at RootCon and they used a slide when they demonstrated how easy it was to distribute malware using DNS. It's just another bastardized use of DNS, much like DNS blacklists were. And I checked with DWarren and these numbers are pretty accurate.

And basically 11 million open resolvers and 380,000 admitters out there. Open admitters mean they can actually see the source addresses of where the recursor resolvers are sending their packets. If I need 3,000 servers, I could probably find 3,000 really easy out of that whole set. Next slide.

Why do we really need open recursors? I heard someone I was working with who was complaining to us about this problem. I said, "Well, why are you running an open recursor?" "Well, you know, sometimes we have some of our customers, they're on our network but then they go behind their office networks; they can't reach our recursers; where there's firewalls, or whatever, so we just leave it open."

Well, there are a whole bunch of services out there, you know, open DNS, Google, Level 3. You know, I love my old Verizon servers, I like using them a lot. So that doesn't hold as much, I think, as it used to. Next slide.

Any filtering – this is interesting. I just tried this today. Four different queries for the same thing – ISC.orgANY. You'll notice that two of these are very small – 258 byte response; 140 byte response. Essentially they're stripping out the RRSigs and all the other interesting stuff that really blows it up.

So when you have a manage service, you might be able to do some steps to mitigate. We'll talk about mitigate in a further slide. But

look down toward the bottom. There are a couple of other providers who are doing 2,000 byte plus responses. Next slide.

There has to be education. Campaigns going out to your [nanogs] and your RIPEs and your apricots to try to educate people about the problem, maybe help people to understand that they want to work with and buy service from BCP 38 compliant networks, as well as avoid networks that aren't compliant.

I'm not certainly going to be able to do that on my own, but there can be a lot of educational materials we can put about this problem and why it's really not just for DNS purposes, but all sources and packet spoofing and all sorts of other methods that it will really help clean up parts of the internet.

Another thing that need to be done is to turn off or modify open recursors. That was a very large number before. That number needs to come down; otherwise, we're going to have that constant threat, that constant gun pointed at any of our pieces of infrastructure. Can the root withstand an attack from 300 open recursive servers? In some way, maybe. I don't know. I'm not looking forward to it. So there are a whole bunch of things that you can do.

You can do some rate limiting on your inbound queries, you can make sure that you're at least monitoring your name servers to see whether you're being attacked and whether you can react to it; you

can implement some scripts or some programs or actually modify your software if you're a product developer to do some reactive filtering. So it's all sorts of things to look at.

Another thing maybe would be if you remember, we had spam, we had open relays. If there are actually things that helped mitigate once you know it's an open resolver, say, "You know what? Maybe I shouldn't accept a full blast packet stream from your IP addresses for DNS," and have a way to filter that, as well as doing some rate limiting from known open resolvers where you might actually be able to configure this just to protect the service for your network. Next slide.

One of the things that you need to be able to do to track this down and actually go after bad guys and have a technique called back tracing where actually going from the place where you have the recursor resolver and you're heading up toward the source. So you go to your upstream provider and your upstream provider goes to wherever that came from and from that came from down to a downstream provider where the source server actually is.

There's a lot of work and coordination that needs to be done. It can only be done in real time. That is, the attacks, they may only last five minutes, an hour, maybe a few hours, if you're lucky. And there has to be coordinated efforts within the operational ISP communities to go after that.

A lot of the people who are really good at this – I'll call them samurais here – they're really good at it but the information in their head – I've been told this by an associate of mine – that it may not actually get out into the knocks, it may not actually stay with ISP. When they leave, then the skill sets go with them and ISP might be left basically trying to hope that they don't get D Doses.

Seven ps - prior planning and preparation prevents piss-poor performance. This isn't something that you can just say, "Oh, I've got a D Dos. Alright, let's figure out how to do the back tracing." There aren't really good how-tos on how to do this for the novices. You have to really have practice doing this to actually be effective. Next slide.

So for ISC, what we're doing with this ISC.org/any is we don't really know yet what the source is. We haven't been able to back trace this to a single source yet. We were basically saying, "Hey, maybe there's some bad customer premises equipment," you know, maybe a malware we'd like to be able to attribute to our particular one. It could be just people hosting services. We're working towards tracking it back to the source.

At some point we're going to blog about the problem, give an FAQ for the people who complained to us in asking what's going on. People really do like working with us after we tell them what's happening. Give them some recommendations; perhaps give them a sensor monitoring tool kit so that they can actually look below

their cursor and actually help create a real-time feed to say, "Here's the packet flows that are actually coming through."

And then we're going to be instrumenting our own name servers so that you can actually see a real-time list here of all the cursors that are coming in or all the cursor servers that are actually being used. And we're going to work with some people out of a Rolodex to actually go after the packets.

And eventually, hopefully, we will get to the source. We will have some data. If you actually are interested in looking at some of the data, we can make it available to OARC. But I'll just give you a quick little example here. Here's a packet tracer. I just took a look at; here's the stuff that's coming in.

They're all coming from the same source; it's not really in a fast flood, but every few seconds you get a different query. It's not really hurting us but that name server – that 129, that 250 – they're feeling some pain because someone's just keep asking them again and again for their ANYs and I just want to go down.

It's basically just showing me that there's the name, that's the name for that name server, that IP address and sure enough, I can go through and query a name through that name server from this venue. It's just another over a cursor. So at this point in time, or 30 minutes ago, I could give that address and put a snort rule and

send it out to a whole bunch of people and say, "Hey, let's go and figure out where these flows are coming from." Questions?

Roy Arends: My name is Roy Arends from Nominet. I knew about this; this is fairly old news; it's a big problem and it needs to be addressed. However, there is a completely new class of attack out there. It's an attack where people register a domain and they list about 10 or 13 NS records and on the right-hand side of this NS records are 13 unique host names.

And somehow all these 13 unique host names have the same address. You send one query to a resolver; the resolver tries to resolve all 13 because the first one fails, the second one fails, the third one fails. Meanwhile, it's sending 13 queries for this amplification taken by 13 to the victim. And of course, the victim address is the address listed on the right-hand side of all these host names.

Then there's an even stronger attack than this one. The 13 name servers listed can actually point to valid names in new or in different top level domains and then you have, in all these individual new different top level domains, you have again 13 host names. So you have 13 squared queries going to this victim. That's 13 squared – that's 169 amplification attack of packets, not zone side, not packet size, but packets.

There's something I've seen once or twice. I've tried to understand what's going on and then I realized this is an out of service attack. So this is a whole new kind of race. Just thought I'd mention it here.

Eric Ziegast: Yeah, it's just fortunate for us at least because we're interested in this, that this attack, actually we have some visibility into it. In the attack that you described, you may not be involved at all; you're just seeing those packets on the network.

Eventually, if we get close to these guys, they're just probably going to change the name or use something else and they'll scurry like cockroaches back into the dark. But there's a chance that we may be able to go after them just cause we know how they're using it.

Jason Fesler: Jason Fesler. Have you brought up your idea of distributed snort rules to mailing lists like NSPsec?

Eric Ziegast: Answer is yes, it's on the last slide. I didn't have the monitoring set before today for this presentation, it's a little bit under time pressure here. I'm working with a particular CCert; I'm working with Cert who's going to be volunteering to take the real-time feed

and turn those into snort rules that will be made available to NSPsec and others.

So when we're gearing up the instrumentation for this, we can actually do this live and automated and not just, "Hey, can I go ahead and give you a call?" and, "Alright, let's go call him in Europe." Try to actually get all the players in one place. That's kind of a little bit what I do with Security Information Exchange – just try to help get all those people working together in real time. Alright, that's it.

Eberhard Lisse: Thanks, Eric. Alright, next up we have Paul Hoffman.

Paul Hoffman: I'm Paul Hoffman and I am here to present what I hope is not a terribly interesting discussion, and that is because one of the conclusions I'll be happy with is this doesn't have an operational effect on the DNS. However, there are some ways that this might get deployed where it does, so I wanted to bring this to you folks early. Next slide.

So very quickly I'm going to explain what DANE TLSA is, so for those of you who don't recognize it, we've got two new sets of four-letter acronyms. Show you what the resource records will probably look like and then talk a little bit about when you might

be seeing these on the net and what the operational effects might be.

So there's a new working group in the ITF called DANE. In fact, both of the Chairs are here – Warren and Ondrej are the Chairs. I am the document author or a document co-author; I am not a Chair. Ondrej and Warren are the Co-Chairs. And basically the problem that we're trying to deal with in the Working Group is people want to start up TLS without having to go to a certificate authority and pay the money and have everyone trusting things that really aren't trustable and exposing themselves and things like that.

So the way to do that is to put your trust of the certificate into the DNS. That is, if I get the certificate from the DNS, I don't have to trust some certificate authority who I've never heard of and for those of you who are familiar with the problem, there's over 100 of these certificate authorities in your browser that you're already trusting. And if people get wise to this, they're going to try to pare it down.

And so the basic idea is, "Hey, if I'm going to be asking the DNS for how to get to this website that is running TLS, I can ask the DNS what certificate, what public key should I expect there." So that's basically what we're trying to do in the Working Group.

Now, notice the last bullet. This requires DNSSEC. Requires is too strong of a word here. It 'should' use DNSSEC. It doesn't

require it at all in the same sense that DNSSEC is not required to get an A record. But if you have a man in the middle who can spoof answers to DNS, just in the same way they could spoof an answer to an A record, they could spoof an answer to the certificate request. It is identical.

So the model that's being used in the Working Group right now with TLS is anything that you can do without DNSSEC would be better done with DNSSEC. But since for a large value of no one is using DNSSEC today, it should not make things much worse. Next.

So a lot of people say, "We're already doing this." Well, actually, we're not doing it for TLS. We've done it for SSH, we've done it for IPSEC and when I say we've done it, there are types that are defined already, essentially no deployment. Well, well, well under 1% for either SSH or IPSEC. Even though it's built into lots of the software, the actual number of people who have started to use those RR types as a way of bootstrapping either SSH or IPSEC is near zero.

I'm wearing the hat that you might have seen on the first slide; I run the VPM Consortium. We are one of the main interoperability testers for the IPSEC market and we never see this being requested, which is not to say that none of my members have implemented it. But the ones who have implemented it have found

so few customers who care about this, they don't even care about conformance or interoperability testing.

However, HTP over TLS seems to be an awfully popular protocol. And so, in fact, a lot of people really want this and for whatever reason, the DANE Working Group has gotten a huge amount of interest in the ITF, more than many of us actually expected. Next.

So now I'm going to describe what was in the draft a couple weeks ago when I turned in this presentation. Some of this is wrong and I'll point it out to you when I get to it. But the TLSA record is going to be requested by you pick the port number on which the SSL service is running and the kind of things – OTCP – you might have UDP if DTLS gets implemented at all.

And they're asking for a certificate record and I'll explain what they get back in the next slide. But we expect at this point that most requests will get one record back. However, more than one record is definitely allowed. There are some models where they might actually want to give back multiple records. But those models usually involve people who would be giving back 10 or 20 certificates which is huge for them anyway. And they can get around that by doing something more intelligent, namely to give their certificate authorities record.

So we don't know that all requests will come back with one response. More than one response is allowed and we really don't

know. Our guess is almost all requests will come back with one response. Next.

So the record that comes back has a certificate type that's only one byte long and the certificate type – there are two certificate types that are defined. One is the end entity certificate, the certificate of the TLS server that you're going to. And the other certificate type is the CA that that certificate you get from TLS is going to chain to. Those are two very different business models. They're both actually needed and different people care about one or the other, and not many people actually care about both of them at the same time.

So you have one octet saying, "Is this the end entity certificate I'm giving you," - namely the certificate of the website that you're probably interested in, - "or is this the certificate of the CA who that's going to chain up to?" In both cases you don't have to look in your giant set of root certificates because if you really care about CAs you'll look for the second type.

And then there's another byte for whether the response you're getting is a hash response or the full certificate. And then the last part of the response is, in fact, either the full certificate or the hash, depending on what that second byte told you. Next.

So the reason why I'm here is that, depending on the choices made in those first two bytes, the operational bit will be different. If, in

fact, what's coming back to you is a hash, it's going to be under 100 bytes. This is just a new record type that has some information, probably not of interest at operationally. Yes, it's a new RR type, but there's lots of those. Knowing that there's a new RR type that might become popular that has responses of about 100 bytes or less isn't that big of a deal to folks in the room.

However, if they're passing back the certificate itself instead of the hash of the certificate – and there are good reasons to do that – you will certainly start seeing things that are over the somewhat magic 512 byte limit. And if you're getting multiple records back in a response, you will, in fact, start seeing things that go over the somewhat magical 1240 or 1280, whatever, limit. And that does have operational impact on some of you.

So, again, we have no idea – let's say this is deployed tomorrow, which it won't be – but let's say it's deployed and people love it and start using it, we don't have a feeling for how many people are going to send back hash types, things you don't really care about. It's essentially like a long test record, or sending back certificates which, in fact, as we heard earlier this morning, 512 seems to mean something to a lot of people who have, for example, Pix Firewalls, things like that. Next one.

So, if people are going to do end entity certificates, that is, here is the certificate of the site here you just asked about, it's likely they're going to use hashes and there's two reasons for that. One

is why not reduce the amount of pain that they're putting on their own DNS servers; and two, there are rumors about this 512 byte thing so why even come anywhere close to it.

However, many people, in fact, either don't want to or are not allowed to respond saying, "This is my certificate; just trust it." They really need to say, "You need to trust this certificate authority who I have bowed to and paid some money to," and such like that.

In that case, it actually makes much, much more sense for them to send the entire CA certificate than a hash of the CA certificate because if you get a hash, you don't know what it is. You're essentially saying, "Here's a CA certificate that's already in your root storer, so you know which one to go to, but it's not really saving anybody anything."

So for the people who want to send back a CA certificate, it's very likely they're going to send back an unhashed CA certificate. Those, again, will be bigger than 512 and some of them are, in fact, just natively bigger than 1024 as it is because some CAs who have bigger lawyers than others who say, "We need to put every disclaimer right in our own CA certificate," have made these giant CA certificates.

It doesn't really matter to most people before today because that just comes in their web browser – not a big deal if it's even a K or

two. However, it does matter if we're starting to send that over the net.

There was a meeting a couple of weeks ago in Phoenix of CA vendors. I call them vendors because I don't really consider the service they provide to be a real service to people, which they had specifically invited a bunch of DNSSEC folks to the meeting to talk about this as well as some other initiatives that at least one CA vendor is starting to see if there was interest.

There was interest during the meeting during the day; there has been absolutely no follow-up. They set up a mailing list that has had essentially no traffic. Part of the problem is CAs don't understand this at all. They do not understand that assigned root zone for the DNS is really like a certificate authority for everything at one level underneath it, and that 256 or however many TLDs we have this year versus next year, are CAs for the next level down, such like that.

The CA vendors just didn't get it. When they do get it, they may want to get more involved in this; we don't know. And if they do get more involved, they're going to be pushing their customers to be sending out CA certificates, not end entity certificates, because there's no reason for them to exist if you haven't gotten a CA certificate from them. So in that case again, the operational impact will be you'll start seeing a whole bunch more responses that are inherently over 512 bytes.

Really quickly, the DANE Working Group is still working on this. I just put out a new draft, literally last night which invalidated some of the text that was earlier. But we're getting to about the same place here. It would be nice if we were finished this summer, and we might be. Every time a new open issue comes up, we have good discussion on it.

But unlike many IETF working groups, we haven't started reopening old issues, issues that have already been closed. So we might actually finish up on time. There are definitely browser implementers who would love this; they're already starting to put code in – experimental code cause we don't even have a code point yet – but are really interested in this because there is a question today with some browser vendors of when they put a CA in the browser, you know, they say, "Well, why did they put it in, why should all their customers trust this?" And some CAs have turned out to be inherently trustworthy. Some CAs have done nasty things.

Well, no one's come back and sued a browser vendor yet that I know of, but there are lawyers out there in the world and the browser vendors have a lot of money – you know, like Microsoft Eagle, things like that – so it's only a matter of time. And so the browser vendors would like to get out of the business of saying, "You need to trust all 150 of these," in a way that doesn't get them sued by the first one who they said, "but don't trust him."

So there is a lot of interest in this; you will see deployment. We have not dealt with the issue that really for security you should have this covered by DNSSEC because we're talking about applications here. It's your browser that would be kicking off this request. Your browser has no idea if the answer it gets back from the DNS was covered by DNS.

This has come up a bit in the DNSX Working Group and was shot down so it has not gone anywhere. It will come up again over time and, of course, if we really do believe that these need to be covered with DNSSEC, then again, the operational impact will be much larger responses coming back.

And then just the last bullet is, we've been only talking about HTTP, TLS – I've also just authored a draft for SMIME - you know the long-forgotten security protocol for email which would do almost exactly the same thing. I could give this presentation almost word-for-word except you would look it up differently.

But it will have the same impact. It won't have the same impact on you cause, of course, no one uses SMIME or we're five years away from anyone significantly using SMIME, but you might see this later on. And that's it. So, questions.

George Michaelson: George. Can you explain why you're simultaneously pursuing application by application specific stuff and also KIDNS or have I fallen asleep at the wheel and there is a meta plan?

Paul Hoffman: The KIDNS is DANE.

George Michaelson: Has it got renamed?

Paul Hoffman: And to make it more confusing, the mailing list is called KeyAssure. So the ITF mailing list is KeyAssure@IETF.org. The Working Group is DANE, but we thought it was going to be called KIDNS.

George Michaelson: So that's the first part. The second part is you tickled my interest because for a very, very long time now it has irritated the bejesus out of me, but when the ops go and change the operating system platform on a remote machine I care about, I have a two keystroke "yes, yes" pattern and then an edit an SSH pattern and then a reconnect pattern that could have been solved in the DNS. When I go read about it, it's even one of the nice people in this room who posted to the open SSH list saying, "Guys, why don't we fix this."

Paul Hoffman: Right. Well, SSH again has fixed it. They have a sur-type. It just isn't being used very much.

George Michaelson: We should be using this. That is such an irritating thing to have to do.

Paul Hoffman: Well, my feeling is that, in fact, once DANE is done and people start doing it for TLS, then they're going to start going, "Well, we should do this for SSH." "Oh, we did do it for SSH. Why aren't we using it?" and it will, in fact, get used. Now, again, a note on operations in SSH is the fingerprint each time; it is never the full cert. So from a DNS standpoint, you're not going to have the operational impact cause it's always going to be under 512.

Paul Cox: Hi, Paul Cox. I just wanted to point out that there are right now patches for NSS, really small patches that you can use if you use (inaudible) NSS which is what Firefox uses. You can actually get DANE validated certificates in your browser right now so you can actually play with this code and it's using the Draft 5 Syntax.

Paul Hoffman: We were surprised at how much interest there was in this. We thought this was going to be sort of another security geeky thing that might pique some people's interest, but the mailing list has

been very active, both with DNS people and with web security people.

Male 3: Hi. I would like to know if the charter for this Working Group has anything to do with the recent hijacks of the internet traffic because there are various certificate authorities in different countries. If yes, why you are still supporting CA?

Paul Hoffman: So one of the reasons that many people are interested in the DANE Working Group is to prevent hijacks where a governmental authority takes over a CA and starts injecting bad data. That's not my interest, but there is definitely an interest in that.

In order to work to that attack – and I do consider it an attack even though it's coming from the government – to work in DANE, they would have to take over DNSSEC or in the absence of DNSSEC, they would have to be able to act as a man in the middle. They could do that today with A records. This makes it a little bit harder, but not significantly so.

And again, I don't want to have this be a discussion of DANE. It's just another use of the DNS. The main reason I wanted to give this presentation is so of those of those of you who care about, "When are we going to start seeing responses greater than 512 bytes?" This will be that. Great. Thank you.

Eberhard Lisse: Next up we have Antonio.

Antonio Cansado: Hi, so I'm Antonio Cansado, first of all, thanks. This is my first time here in OARC. I'll present to you on just work with some colleagues of me. In fact, this is not my major contribution. I am mostly presenting what they have been doing. This talk presents a Threshold Cryptographic Backend for DNSSEC and DADS that it can connect with existent platforms for DNSSEC and provide a distributed backend for the cryptographic part.

I put it in that words because it's still on-going work. So first of all, who are we? We are the research labs of NIC Chile which has been founded by NIC Chile in 2008 and what we want to do is to transfer some technology to the regional industry mostly.

So most DNSSEC architectures look like this. They have some engine that generates the DNS zone based on some database with these zones. For that they rely on a signer robot sometimes that needs to work with a cryptographic provider. So what typically implementations do just like combine with DNSSEC, they do something like this connecting to a cryptographic backend with PKCS 11. And the one thing that (inaudible) is that zones need to be resigned periodically since they have some expiration date.

So for that the signing robot needs to access some cryptographic backend, usually some hardware security modules. And usually keys within these HSMs cannot be cloned. This is not entirely true. You can have a cryptographic token and replicate the keys, but they are not meant to be cloned.

But when you need to replicate your server you'll have to do that. You can clone the keys or you can create a new key and have your parent zone know both keys. So your parent key, this is okay, but what work I am going to present to you here is some alternative to this key.

So why is that? Because as Roy Arends showed in a previous presentation, although it's kind of hard to break these things, they eventually do. Maybe it's not a hardware failure, just some natural disaster. So you don't want to have the key stored in a single place. So, again, what we're going to do is take a look of what can we do with this.

So starting from the same architecture as shown before, what we do is instead of having a single node that provides us our cryptographic facilities, we do it with a set of nodes. And what does this mean is that we have a distributed architecture, that the backend is implemented by any number of nodes you may want and what we do is that the private key is actually split into shares and distributed among these end nodes.

This system is full torrent in the sense that even if some of the nodes fail, the system can continue working on without system disruption. It is robust. And for that failures and attacks are mitigated by implementing nodes in different programming languages and operating systems. And one of the good things is that no one knows the complete private key and we will demand that K nodes less than [N] must sign the petition in order to avoid some fake signatures.

So what we have done is we've taken a paper by Victor Shoup presented in the year 2000 that's called "Practical Threshold Signatures" and what it shows is how RSA can provide a threshold signature system, NK. And what that means is basically that the private key that is divided among N peers so they have pieces of this private key and just KPs are needed to create a signature. So these K entities sign pieces of what you need and afterwards you can validate if those shares are correct with a public key. And there are also somebody there that I've omitted in here.

And this approach we believe fits well with DNSSEC because it allows us to deal with systems with high volume of signatures. So just to have an idea of how the work flow looks like, so let's take a node and call it Signature Dealer. If you want to sign something, the Signature Dealer will send it to your backend.

In here we've implemented with some keys but just to show you that this can be done asynchronously, suppose that just three of the

five nodes are alive and take the responsibility of signing these things. And one day, once they have signed, they didn't receive the answers, valids if the shares are okay or not and they it will send a signed entry to the men that sent the request.

It is worth saying here that we usually demand that more than half of the nodes are alive just to our working and are not corrupt and to guarantee that the system has not been compromised. So this is still a work in progress. We have prototyped these things but we have not provided any benchmarks yet, so sorry for that.

And we also are working on two different challenges. One of them is we believe that there is some bottleneck at the Signature Dealer and that in fact, that one also represents a single point of failure that is against completely what I have been talking here. So our solution for that is changing the architecture to appear to be one where every node that I've shown before is actually a dealer.

Another benefit of this approach is that it naturally implements some load balancing and there is no single point of failing. Another challenge is that – it is a bit hidden in here – is that in fact the private keys at first for bootstrap, it is created as a whole and afterward it is divided and sent to the peers and then destroyed.

And for a moment, it is actually created entirely. And for that we have seen a recent paper published in 2010 – sorry, but I have it in my presenter notes that I can see in here, so I can't tell you exactly

but I'll show you afterwards. And paper shows a way to create private keys in an entirely distributed way and this way shares are created directly at the target nodes and nobody ever knows the complete key.

So just a brief summary of our approach. I presented you a distributed cryptographic backend for DNSSEC. We believe it's not limited to but we believe it fits well in this context. In some ways can replace Hardware Security Models or just complement them. How is that? Because every node that we use could have an HSM in their own, but no one will ever have the complete key. It can also be integrated with all the DNSSEC engines – an open DNSSEC, for example.

The advantages is that it's distributed; it's robust; it's low cost; it's a mostly software approach, but more importantly, nobody holds the complete key. And another thing is that risks or failures can be bounded as you want. If you want to lower your risk, just put another node and the total failure rate is lower.

The disadvantages that are a weak point in here is authentication with the dealer. Okay we have just pushed the problem to the dealer instead of leading it back in, so there is where we are looking at right now.

Another thing is that the number of shares is fixed upon key creation. What that means is that if we use 10 nodes for our

system, if we want to use 20, we'll have to recreate the key. Or get all the shares in what was the complete key and then redistribute it once over.

It's slower than typical RSA. I don't have the figures in here so you'll have to believe me for that. And some bandwidth is used, whereas when you have just a local hardware you don't have any bandwidth and here we have some. So that's it. Questions? In my webpage I put it in there and you can see more information. Sorry it's currently being translated into English. Right now it's only in Spanish.

Ondrej Sury: First, thanks for presentations. My name is Ondrej Sury from CZ nic. Thanks for presentation because that's exactly what we are speaking with the other Ondrej in the context of the root zone and here's the question. At least some of it could be used to sign the root zone, especially the distribution of the RSA so nobody holds the complete key.

That would be excellent but more power to the community that the key shareholders or the (inaudible) officers in the root zone process will actually hold part of the keys so the key is not held by ICANN, but in the community. That would be really great.

Antoin Verschuren: Thanks for the presentation. We have something – my name's Antoin Verschuren, SIDN. We have something similar in place,

but not for the distribution of keys. And I don't believe that that's actually something that's necessary because if you can get on a signer engine, you'll need to do an operation which signs with the complete key.

So distributing them over multiple machines doesn't give you anything more security because it's the line between the signer and your pool with the keys, that's only one line.

It does, however, prevent from hardware failures and that is something that we've done as well. We have HSMs which are in a pool just for hardware failures, but not for distribution of keys.

Antonio Cansado: Just something to complement what I said. If you see the second formula in here, it's we have what (inaudible) calls verification keys that you can verify if the shares are compromised or not, if they are correct. It's a public key for each one of the shares so I think – I'm not sure if I'm sure if I missed your question, but I think it's related – you can always verify with a public key and you have as well some verification keys local for each one of the nodes. I'm not sure if I missed your question.

Paul Hoffman: Paul Hoffman. A lot of work has been done since you published the paper and when I say a lot of work, I don't just mean the cryptography, but actually interesting stuff on how to split the

keys. One of the things that has happened in the last couple of years, especially if you read some of the Chinese cryptographic literature, is the realization that if what you're really trying to do is just to – you're going to trust everybody, but you want redundancy – you can actually just duplicate some of those nodes.

You said it's a problem if you go from 10 to 20 you have to split the keys. You don't actually have to; you can just duplicate those. Similarly, you were saying you were a little bit concerned about the dealer cause now that's a simple point of failure. Maybe we do a P to P dealer which, in fact, I don't think will work cryptographically because your PKCS 11s will all have to have as many signatures as other…

You could just duplicate the dealer again so every request to signed zone actually goes to two dealers, they duplicate all of that… It doesn't matter. We're talking about a second's worth of processing time for somebody. And then either dealer gets a single answer, can come up with a valid answer.

It won't even be the same answer cause it will come at different times and such like that, but that that's a legitimate signature. Really you should be deciding do you care much more about redundancy and therefore lack of failures or much more about security of the individual owners. When Ondrej was talking about, "Oh well, we have all these owners of the root," that's a very

different question, and your answers could be much, much simpler if you're just trying to do this for redundancy.

Antonio Cansado: Yeah, I totally agree with what you said. In fact, most of the things you said are the approaches that we're looking for right now. On the single point (inaudible), the most easy solution is just to have a first row of Signature Dealers that you can access just like you said. And the other one's the Peer to Peer architecture and we have to complement it with some kind of voting system so if some of the nodes are compromised, they can detect each other and then say, "Hey, that one is compromised." And instead of just asking one node, you ask two or three or four and then you get rid of that problem, so yeah, exactly.

Stephane Bortzmeyer: Stephane Bortzmeyer from AFNIC. Maybe I missed something, but it's not clear what is the current state of the prototype. Do you have at least one zone which is currently signed with the system?

Antonio Cansado: No, this is just a prototype in our lab. What is working right now is mostly what you can see in this figure. We have to decide which node will act as a Signature Dealer and that will have all the nodes that can sign things. We're also working on a signature robot that will create the NSEC registries for us, but that's a different work.

And we plan to distribute this code as a GPL or similar license and we're just packaging and make sure that everything goes smoothly right now.

Sebastian Castro: Hi. Sebastian Castro would like to know how do you protect the key shares and what prevents an attacker to collect enough shares to recreate the private key?

Antonio Cansado: How to protect the private keys right now? We can say that we plan to implement these things in different platforms in the system and if you really need to protect these keys and you can afford it, you can always buy some hardware to monitor each one of these nodes and put the key inside them. So we have those two approaches in mind.

Eberhard Lisse: Thank you very much. And last up we have Dave who's giving us some beer so I'm sure he'll do the job.

Dave Knight: Hello. My name is Dave Knight. I do Operations, ICANN. This was just a little something that came out of some work in the last few weeks. During the transition of (inaudible).ARPA from root servers to the new set of servers, we wanted to do a mini day in the

life data capture of queries arriving at the outgoing root servers and the incoming new IANA and ICANN operating servers.

I've been having a bit of experience in the past of trying to get people at short notice to do widely deployed data capture. I know that it can often be difficult to convince people to run software that they don't already have installed. So I had to look at how we might do this using TCP Dump.

So I'm well aware that there are better ways to capture then filter DNS packets than TCP Dump, but they might not be available for the platform you're using, site policy might forbid the installation of software – I've encountered that before in a support situation – and this method using TCP Dump actually be faster.

So I guess everyone's familiar using TCP Dump with a pcap expression like this to look at a conversation between a server and a cache. To match specific names, we need to match against the DNS wire format. So looking at the format of a message – in this I'm only looking at matching in the question section, so really the only important thing to know here is that the head of section is 20 bytes long here so that's where you're going to start looking at things.

The question section looks like this. It's a variable length QNAME, and then the QTYPE which is two bytes long and the QCLASS which is again two bytes long. Pcap exposes a raise of

the EDP portion of the packet TCP and so it's into those that we'll look for names.

DNS is case sub-sensitive; pcap can't be. So when comparing a name in a packet, we look at each byte and we look for upper and lower case version of the string. So that's what the query name, the query type, the query class, ARPA,/NS class N looks like in the EDP portion of a packet, starting 20 bytes and first there's the label length, then ARPA, then the known label for the root, then two bytes for the query type and two bytes for the class.

But the query name is variable length so that pattern there will only match exactly ARPA dot. To match things under dot ARPA, we could write that same expression at incremental offsets then that array 255 times and then we can match everything that's underneath the ARPA. This creates quite a big pcap expression.

So that's what that would actually look like and then imagine that 255 times. That only matches IPV4UDP so we can do this for TCP and for IP6UDP and IP6TCP. Now in pcap, there is no arrays exposed for IP6 UDP or TCP, so we're just looking in the IP6 array which is a level above that directly, which leads into the optimization for this, which is not to think about UDP or TCP at all, but to look directly into the IP and IP6 arrays and just overlap that such that you catch any possible positions for where this might be in the EDP sections or the TCP sections of the packet. That optimization gets you down matching for ARPA to an expression

which is about 7,000 lines long in that format on a couple of slides back.

You can go one step further than this and match in the ether array but this does make things a bit more complicated because then this will only work when your capture device is Ethernet. Manually adding these expressions is probably a bit problematic so I've written a little script to generate that. You can go grab that – it's on a webpage.

And I've done a very little bit of performance comparison with DNSCAP. DNSCAP can do much the same thing but it can do it with regular expression. I took 50 million queries captured in L root mode and if I find the first 50,000 matches for in-addr.ARPA, in that using DNSCAP and this method, DNSCAP can do that in under a second. This method takes over five seconds.

However, if I find the first 5 million matches, this method is quicker. DNSCAP takes about a minute and a half to do that; this takes slightly under a minute. And the reason for that is there's an overhead of about five seconds, at least on my Mac, where TCP Dump compiles this expression. It takes about five seconds to compile an 8,000 line pcap expression.

At the bottom there I've got a compare it and file sizes of what comes out in that. DNSCAP is stripping the link (inaudible) information so that file is smaller. Also I noticed a very slight

difference in what is captured with this method versus DNSCAP. I seem to get 10 messages more in 5 million than DNSCAP. I'm not sure why that is.

This is only capturing the particular TCP packet which has a match in it then after the session is ignored and I haven't even thought about the implications for fragments, but I don't imagine that I would ever see a fragment in question so I haven't worried about it.

So, yeah, this might be useful to you because you already have TCP Dump installed and you don't have to install anything else because you can run the script to generate the expression elsewhere. And it could be fast if you have large data sets to process and potentially it's also less resource intense of using it in a live capture on a name server situation, but I haven't actually tested that.

Jeff Sisson helped debug some of this stuff and I was originally inspired to look at this from a presentation that Shane Kerr did at RIPE 56: "TCP Dump DNS Built the Rules For Fun and Profit." Any questions?

Roy Arends: I am Roy Arends, Nominet, still. I got a question. If I understand correctly, your script actually makes several iterations because the sub-domains below ARPA can be any length and so you basically

iterate over that in all possible places. Isn't it much more simpler than every domain name you get in the packets you basically revert it and then match it, right?

Dave Knight: Potentially. It's not something that… I'm not very familiar with what I can do with pcap. This was just what I give on a TCP dump command line and didn't really think about it more than that.

Roy Arends: Oh, okay. If you, for instance, think about raw (inaudible) data, the trick that I just described is trivial. You basically, so instead of- In dash other… By the way, the person who actually thought of the idea to first have a domain name and then a QType and QClass, right, next time you design a protocol, have the fixed field first and then the rival fields. Anyway, the trick is basically ARPA does in-editor. so you can easily match all the sub-domains if you want to. It's two or three lines of C to write it. Just an observation.

Dave Knight: Before you began I did make the disclaimer that not all our (inaudible) tools were available so you've beaten me-

[background conversation]

Eberhard Lisse: The authors of DNSCAP have questions. Is it possible that, due to the defects or limitations in DNSCAP, which you discovered and highlighted here, that you will be sending patches?

Dave Knight: I'll send comments.

Eberhard Lisse: And did you compare those at all utility-wise against other INCAP or N message?

Dave Knight: I haven't. Like I say, I did this, it was more of it purely by how do we solve this problem without installing any other stuff, so looking at any other stuff was very much outside the scope.

DWarren Russells: Hi, I'm DWarren Russells and I'm one of the co-authors of DNSCAP, I guess. I was just going to tease you a little bit about some of the other things DNSCAP can do like compress your files and upload them to OARC and other things like that.

But one thing that might be actually interesting is you could combine these two, right? You could give this huge filter to DNSCAP and see how that performs. In all your free time you can do that now.

Eberhard Lisse: Anybody else? No? Then I think we are done for the day. Thank you very much, everybody. Enjoy your beer. Thanks to our presenters. And we'll start up again tomorrow morning. I think it's 9:30 and another fun-filled day.

[End of Transcript]