ICANN74 | Policy Forum – Tech Day (2 of 3)
Monday, June 13, 2022 – 15:00 to 16:00 AMS

UNIDENTIFIED FEMALE: Hello and welcome to Tech Day part 2. Please note the session is being recorded and governed by the ICANN expected standards of behavior. During this session questions or comments submitted in chat will be read aloud in the proper form as noted in the chat. If you are on Zoom, you need to be on Zoom either remote or in the room. If you have a question, please just raise your hand. When called upon, unmute your microphone. For the benefit of other participants please state your name for the record and speak at a reasonable pace.

With that, I will turn it over to Roy.

ROY ARENDS: Thank you.

EBERHARD LISSE: Can I just quickly check? Can you hear me?

ROY ARENDS: Yes.

UNIDENTIFIED FEMALE: That sounds much better. Thank you.

EBERHARD LISSE: Okay, thanks. It's always good to have backup.

ROY ARENDS: All right, thank you for accepting this presentation. This is a short work, something fun that I did a while back, and is basically an Internet wide scan of root-hints. Next slide, please.

So a quick introduction. What are root-hints? Those are the names and IP addresses of authoritative name servers for the root zone so the software can bootstrap the DNS resolution process. Now I didn't come up with that myself. I lifted that from RFC 8499, DNS Terminology.

Many pieces of software like Unbound and PowerDNS and BIND come with this list built in, and this file is often used for priming. And you can find this file at the link on the screen. Here's a caveat though. You can't actually remotely ask for these root-hints because the root-hints are used for the process to prime the cache.

So what I mean by that is what you find eventually in a cache is the result of the root-hints. It's not the actual root-hints. I hope that makes sense. The root servers are asked what they think the root servers are, and so the result of that process is put in cache.

And the scan data is actually a scan for those records. It's a slightly different interpretation of root-hints that I'm using here. Next slide, please.

What I'm doing is a scan of the IPv4 address space for SOA records for the root zone. The cool thing about SOA records is they have a little bit more information than just name server and address records. What is mostly returned with SOA records are the NS and A/AAAA in the additional and authority section. But like I said before, these are the result of the priming process and not necessarily the actual root-hints.

So I'm going to ignore these NS and A/AAAA records for now and start just looking at the SOA record. And you will see in a minute that's much more interesting than the A/AAAA records. It's not that I'm not going to do it. It's just that this presentation focuses on the SOA records. All right, and hopefully we can understand what servers are using and resolvers are using as root servers. So what are the expectations? Next slide, please.

For 20 years the SOA RNAMI—that's one of the two strings in the SOA record, the second string in the SOA record, if you will—for the IANA root zone is nstld.verisign-grs.com. Now the RNAME in any zone, in any SOA records in any zone is the domain name representing the administrator's mailbox. It's essentially an email address if you change the first dot into an @ sign.

We expect that the bulk of resolvers that we ask, the bulk of things that return information use the IANA root servers—a..m.root-servers.net, etc.—and will return SOA records with the RNAME containing nstld.verisign-grs.com.

Another interesting artifact in the SOA record is the serial number. What we know is that the convention used as of the current IANA root zone has the SOA serial number of a date plus a version for that date. So we expect the bulk of resolvers that use IANA root servers to actually return that serial number. Maybe at most a day behind because the time to live of this SOA record is 86,400 seconds which one day. On earth at least. All right, next slide, please.

So what's the setup? How do we do this? We send a DNS message, a very simple one. It's a request (QR=0). A QR is a bit in the header. Now this is important later. I'll show you that in a minute. It's for a standard query. Standard query is the actual term. OPCODE is then QUERY. With a single question the QNAME is an empty label also known as the root label, QTYPE is SOA, QCLASS is infinite class. We don't use any extended DNS, so no EDNS. All header bits are 0, including the identifier. The 16-bit identifier is also 0. All right, next slide, please.

So to whom are we going to send this? Well, a simple, naïve approach is to just send it to everything in IPv4 minus multicast, experimental, RFC 1918 spaces, etc. What is better, and a

colleague of mine reminded me of this, is where you select routable addresses from a route view. And so I got this from the University of Oregon route views archive. You basically get a file that you can select the routable addresses from.

And then you remove the DNS-OARC's don't probe list from it. If you talk to DNS-OARC, they have a nice don't probe list. I'm not sure about the maintenance status of this right now, but we've used this to make sure we don't upset anyone too much.

And it's around 80% of the entire IPv4 space. I'm using ZMap to send these queries. ZMap is a tool. Zmap.io if you want to find the software. It uses an allow list which is basically our targets and a block list. That's basically the don't probe list. And a very simple hexadecimal string to represent the DNS query. And you have to specify if you want to send over UDP and then offer to which port.

The reason I'm using a hexadecimal string is because even though ZMap comes with the DNS module, it contains a bug. And if you look at the ZMap.io bug report, you will find a report in there. It's not significant, but you can work around it with that. Also, the DNS module is a little bit too limited. I did not want to send the RD bit, set the RD bit in the header, and the DNS module only sent queries with the RD bit set. So I simply used a hexadecimal string to send queries. All right, next slide, please.

We get to a section that's a little bit more interesting. Statistics, the results. So we sent about 3.4 billion queries. We got 10 million

responses received. That's pretty good. That's a response rate of 0.3%.

It turns out though that about 3 million, that's about a third, right? About 3 million responses had the wrong identifier. The identifier was not 0. Which is strange because we set the identifier to 0 and you expect 0 back.

And an additional half million were duplicates basically coming from the same—what I mean with duplicates is responses coming from the exact same server. Which is also strange because a single server only receives one query from us.

So I investigated this a little bit and in almost all of these cases it's either the host forwarding a message. What I mean with a host here is basically one of these home routers forwarding a message or bouncing it back to a resolver. So if you exclude those, you have about 6 million responses remaining. Next slide, please.

In order to filter those out I've broken this out in different RCODEs. RCODE is the response code. Top of the list is REFUSED. It's also what we expected. Not everyone wants to be queried. So that's about 3.7 million. NOERROR, that's the one we're going to dive into a minute, 2.2 million. The other one I want to highlight is syntax_error. That's something I invented myself. That's what my parsing code couldn't parse, and on inspection there's a lot of rubbish out there. Stuff that comes back is not DNS.

There are a few other codes in there like YXRRSET, NXRRSET. Those have something to do with dynamic updates. But I can tell you the rest of those packets didn't even look like a response to a dynamic update. So it might be that my code was so advanced that it could parse and not be a syntax_error. Or it could be that they just sent something rubbish back which happened to be parsable. All right, let's look into the NOERRORs. Next slide, please.

Here I stumbled upon and interesting bug, if you will. About 875 responses of those 2 million did not have the QR bit set. Which is a little bit strange because I cleared the QR bit implying a request and I expect a response back with the QR bit set. And these did not have the QR bit set which is a problem.

Years ago, 20 years ago, Jacob Schlyter and I found a whole bunch of implementations that did not adhere to this, that did not have the QR bit set. Sorry, if you would send a query and you set the QR bit, it would still respond to it. If you think about this, you can actually have loops of implementations talking to each other if you don't detect this bit or you don't use this bit.

I wrote an IETF draft at the time. It was basically a one- or two-page draft that basically said if you send a request, QR clear. If you send a response, QR set. And check these things. Don't respond to responses, and don't respond to queries with a query.

At the time it was so obvious that implementation shouldn't do this that it didn't get any traction within the IETF. But also, 20 years ago I remember I had a discussion that I had quite recently again. People say, "But it's actually not specified anywhere that a QR bit must be sent in a response." And if you look at RFC 1035 and 1034, it's really not specified that the QR bit must be sent or that these things must be checked. So I'll dust off the old one-pager and send it back to the IETF and see if the working group can be interested in this. Next slide, please.

Sorry for this sidetrack. Let's continue with the results. About 45,000 had the TC bit set. Now that indicates that the response was truncated, but it's more likely a simple denial of service mitigation technique. It's basically if you really, really, really want to know the answer and you're not just using me for a denial of service attack, then please retry over TCP.

Then another 600,000 responses had the RD bit set which is also weird because we never set the RD bit because we don't want any recursion done. We just want to have the information that's already there in the cache.

So just not to bias any of these statistics that we have, we're going to ignore these messages for now as they contain no additional information or they have caused additional recursion. So what we're left with is 1.2 million responses that contain and SOA record. All right, next slide, please.

So this is to me the most interesting slide. What we expected, about 93% have the IANA root configured, nstld.verisign-grs.com. What I mean by configured is either the resolver asking the IANA roots or is actually a server having the IANA root zone.

There are three strings in here that refer to HostGator. There's hostgator.com, hostgator.in, and hostgator.com.br. Now HostGator is obviously a brand name, and I think this is either a hosting provider or some routing thing that does DNS and has this preconfigured.

And the other one interesting in there is hostmaster. And hostmaster is a string, almost all of these 20,000 have to do with PowerDNS. Before this session I asked Peter van Dijk of PowerDNS a little bit more information, and it's basically the default if you don't—correct me if I'm wrong, Peter—it's the default when you don't specify an SOA serial number. This is not a bug. This is all on purpose.

The rest is not as interesting. You can see how thin the longtail is. So let's go to the next slide.

If we break down the MNAME distribution, now MNAME is the first string in that SOA [R data part]. The M stands for primary name server, and it's basically an indication of what the primary name server is. Of all of those listed before, seven did not have a.root-servers.net. Now I'm not going to go into these because they're not statistically significant. I left it on the slide for people to read

if they want to refer to it later. But most of them have a.root-servers.net.

Now the last thing I did—next slide, please—is the SOA serial number distribution. Again, [inaudible]—and I forgot to [inaudible] this year, but it's the [inaudible]—had 1.1 million and had the most recent serial. So the scan was done over a period of a few days. I used a server that's not that quick, and so it had at most one day after the query was sent.

Then in between the below part and the most recent part, we had 406 entries that were in between 2019 October 30 and today. And then this is the longtail, and it's just a curiosity but I thought it was funny. There are some very old still configurations out there. So nothing wrong with that. There's probably no one using them. And if they are, they will see that they can't resolve everything. But, yeah, that's it really. Next slide, please.

So in conclusion, it's actually fairly straightforward. If you ask the world what they think, you get a lot of different responses. Some of them make no sense at all. So it's actually less interesting from a statistics point of view, these kinds of scans. But as a result of the brokenness, and I'm going to revise the QR clarify draft, we now know that stale configurations lead to old versions of root zones. So if someone wants to scan for old versions of root zones, this is the way to do it. It's a lot of work, it costs a lot of time, but you will find them.

And we also see that a large amount of consumer routers forward or bounce DNS requests. Now we know that this is nothing new, but what I've learned is there are various ways of doing this.

And we also noted a large amount of DNS hosting providers use private root-hints. That really doesn't mean anything. It's basically if you ask them for anything else they're configured for, they either respond with refused or they respond with the root-hints. And in this case it's different than the IANA root-hints.

Most implementations use IANA root-hints. And another careful conclusion, and I say this carefully, there's really no diaspora of intentionally alternative root-hints. There are some alternative root-hints, but they are limited to some authoritative servers. But there's really no indication that they are used by a significant set of resolvers. And by significant I mean at least 50 different [AS numbers] that use an alternative root.

One more slide for questions, answers [about] my presentation. If you have any questions, send it to my e-mail address, roy.arends@icann.org. It'll be on the first slide, it'll also be on the list. All right. Thank you.

EBERHARD LISSE: Thank you very much. As usual, quite interesting. One thing comes to mind, one should never ask questions one doesn't know the answer to. Comes up with interesting results. I read from

Warren—who doesn't have the ask question bit handy—the cluster of 2012 and 2016 SOA seems interesting. He wonders if there's an interesting story around there. Maybe a question for the Internet history list to look at what happened in this year.

ROY ARENDS: Yeah. I think it's all very interesting. So I've looked at—every time you look at these things, you get more questions. So I don't know specifically why 2012 and 2016 are that interesting. 2012 may be the introduction of new top-level domains and people refused to have them configured to a reasonable root zone. 2016, I don't know, maybe DNSSEC-related, we rolled a key around that time. Oh, no, that's much later. Sorry. That's not it. But yeah, I'll look into this further. I'm also happy to share this information. I can't really share those IP addresses, but I can relay queries to those IP addresses if you want to. Thanks.

EBERHARD LISSE: And if you get up with some interesting answers, feel welcome to ask us again for a slot. It's always good to have interesting stuff like that. Thank you very much. Now, after the root hint survey, Peter Robberechts will talk to us about some machine learning. I don't see any hands or remarks in the chat. Peter, go ahead. Thank you.

PIETER ROBBERECHTS: To give some context, I'm a PhD student at the KU Leuven in Belgium and this is an ongoing collaboration with DNS Belgium, so the domain provider for the .be domains. And I would like to begin by making this observation about how the detection of domain abuse typically works nowadays.

Imagine that someone takes this domain, like myaccountverify.be which I have shown here, and then they might use it to do something that doesn't add any value to Internet users. Here, they probably use it to try to steal user credentials. But of course, you know that this isn't the entire story.

Typically, a story begins with a domain being registered and then hopefully, it ends with the abuse being detected and the domain getting blocked. So you have this three-part story that goes from registration to use to detection.

And the problem with this story is that the detection is [inaudible] reaction to the use. That means you can only protect users after a certain amount of other users have been exposed to the risk. That is pretty suboptimal, because it means you have this latency between when the domain is first registered and when the user protection takes effect.

So what we want to do is take this story and eliminate the need for the domain to ever be in use. We will try to get to this world

where we can go from registration to detection in a matter of seconds.

And to be able to do that in seconds, you have to automate it. So what we're looking for is an automated system that based on the data we can get from registration can decide whether the domain name registered will be used in malicious activities or not. And if the system thinks that the registration is okay, then the registrant will be able to immediately use the domain name. Otherwise, the human will have to look at a registration manually and maybe then the registrant will have to prove his identity, for example, by sending a copy of his passport. And if he can't do that or doesn't bother to do that, then the domain will never be added to the zone and no people will be harmed.

So in summary, what we try to do is predict whether a domain will be used for malicious or abusive activities before that happens and as early as registration time. And if you think about it, that is quite crazy, but because all the approaches that are in use today maybe except for Nominet's system that was presented some time ago here. But all or most systems that are in use today in some sense rely on the domain being in use. Either they look at the resources that are being put up there and make some observation about that, or they look at the traffic that is generated by users resolving the domain name. But instead, we want to do it at registration time. And actually, that is something that DNS Belgium has already been doing for quite a while. So

currently, they have this policy of delayed delegation where they use a simple rule-based system that looks for suspicious registrations and registrations that are identified by this system are not added to the zone before they are verified. And the rules in the system look for typical keywords that often appear in malicious registrations like [inaudible] and for clear indicators of fake contact info.

So the idea is that abusive registrations have existing properties like the use of typical keywords or fake contact info. But from previous research about the detection of spam domains at time of registration. We also know that they might use similar infrastructure like name server and registers and that there might be certain use patterns. Like they might register domain names with very similar strings in them. So the idea of this presentation is to show you how we train the machine learning model to learn these properties which we evaluated on four years of .be registrations.

So as you know, good machine learning starts with good data. And there we have this database which contains all domain registrations, updates, and expiration events for a period of over slightly more than four years in .be zone. As well as all these registrant verification reports from manually checking the contact details of all registrations. As well as abuse reports both from manual verification and from blacklists.

And to give you an idea of the scale of this dataset, it contains about 1 million unique registrations, about 40,000 registrant verification reports, and about 20,000 abuse reports.

Then from this database I extract what I've called here a DomainEventGroup, but actually this just captures the entire lifecycle of domain from its registration up to its expiration.

And for each of these domain even groups, obviously we have the domain name that was registered, when it was registered, as well as some information about previous registrations like the registrar, the amount of unique registrants in the past.

Then at the end we have this list of events of which the first event and the one that's most relevant to us is a new domain event or a domain registration event. And for this event we have the registrar where it was registered, the contact details of the registrant, and the list of name servers which was linked to that domain at the time of registration.

Then as the contact details we have the name of the registrant, his email address, his street, the city, postal code, phone number, and optionally organization name and VAT number.

So now that you have a feel for the data that we work with, I would like to walk you through five categories of features that we use. And these five categories are based on five underlying assumptions that we make about malicious registrations.

The first assumption is that these malicious registrants are lazy and if they register multiple domain names, they will probably do so with very similar registration details.

Second, since they want to do illegal activities with the domains they register, they probably use fake contact info.

And third, they will probably reuse infrastructure like name servers and registrars.

Then fourth, they will probably reuse domains. What I mean by that is that they will probably use domains that were expired so that they can piggyback on the reputation of the previous owner of that domain, and that makes it easier to avoid blacklisting.

And then fifth, when they register multiple domains, there are probably some similarities between the domains that they register.

So the first set of features looks at the reuse of contact data. And here the idea is to create these blacklists of all registrants names, addresses, email addresses, phone numbers, organization names, and VAT numbers that have been identified as being fake in the past. And then for each new registration, we count how many of the entered contact details appear on these blacklists.

And actually, it appears that from all of our features, it is by far the most discriminative one. So what we found is that 45% of all malicious registrations reuse blacklisted contact details, while it

is only the case for 12% of the benign registrations. And actually, the benign registrations over here are mostly unlabeled examples, so a portion of them might be malicious as well but were simply not identified as being malicious.

So this first set of features mainly looks at or search for contact details for which you already know that they are fake. Then we also have a set of features which try to identify for new contact details whether they might be fake. And here we used three strategies.

First we look at individual fields, and for these individual fields we check for some lexical patterns and keywords which might be indicators of fake data. Like for registrant's name we check with it has vowels, whether it contains duplicate words, we count the number of characters. The same for the registrant's address. For the email address we also check whether the hostname resolves. For the phone number we look at increasing or repeating numbers, and we check whether they have the correct length. And for the organization name, again something very similar.

Then a bit more intelligent we check whether there is consistency between multiple fields. So for example, for the registrant address they have to enter the country and city. We can infer their local time zone and then we can see whether according to that local time zone was the registration made during daytime or during nighttime.

For the registrant email we check whether it overlaps with the registrant's name or organization. For example, if I register a domain name as Pieter Robberechts, then it would be more logical that I provide an email address Pieter.Robberechts@gmail.com than that email address would be KatieSlate@hotmail.com or something like that.

Then finally, we check with some third-party external data. So we look into the GeoNames database. That is a database with all city names and postal codes. And we also have this registry of all Belgian companies, and then we check whether the company that was entered in the registration actually exists in that database as well and whether the address that was entered matches with the address in the database of the Belgian government.

Then as a third predictor we look at the reuse of infrastructure. And here we make two observations. The first one is that most malicious registrations come from a small group of registrars. For example, here you see that for the registrar Netim more than one-third of the registrations that come from that registrar are actually malicious. And overall, we found that more than 50% of all registrations come from only four registrars. And we find something similar for name servers. So again, we see that for some name servers the majority of registrations are malicious.

Then fourth, we look at registration history features. So here we look at the data of previous registrations and mainly at the data of the most recent previous registration. And that gives us features like previous registrar, re-registration latency, so the time that has passed since the previous registration expired, and we can split that up in three categories, whether it was a brand-new registration, so that means there were no previous registrations, whether it was a drop catch registration, that means that re-registration happened less than five minutes after the expiration, or the third category is a retreat registration, so that's a re-registration more than five minutes after the expiration. We also count the number of bad WHOIS cases on past registrations. And actually, here, the most [discriminative] feature of this [set] was whether there have been previous registrations. So 45% of the malicious registrations reused an expired domain compared to only 24% of benign ones.

Fifth, we look for string similarities between domains. This is quite similar to what [inaudible] has done, but a bit simpler. Our approach consists of three steps. First, we count all quadrigrams in benign domains. So you see the quadrigrams which appear most often in benign ones and the one that is most frequently appearing is tion, then we have shop, which is quite understandable in the context of domain names, and basically, this is a language model of domain names in the .be zone.

Then we do the same thing but only for malicious domains. Again, we count how often each quadrigram appears in malicious domains, but we only do this this time for a limited period. We do it for the past 7 days, the past 30 days, and the past year, and that allows us to capture as these patterns evolve over time.

What you see is you actually end up with the same quadrigrams on top, only the distribution changes a bit. But when it gets interesting is if you look at which quadrigrams are overrepresented in the malicious domains. Then you find things like caix and aixa which are slight variations on the name of a Belgian bank. You'll find a lot of quadrigrams which are derived from YouTube, which is obviously very popular in the .be zone because it ends on .be. And then you have a few campaigns, one about Connexus which is a loan provider, one about Calisthenics, campaign about hoses and a campaign about Amazon gift cards.

So those were the features. The next ingredient we need for a machine learning classifier are the labels. So remember, we have about 1 million registrations in our training database, and out of these 1 million registrations, there are about 27,800 labeled as bad WHOIS, which is used to indicate that they provided fake contact details.

Then there are also about 11,000 registrations which were explicitly labeled as good WHOIS but typically, these registrations were suspicious in some sense and when a registrant could then

still provide proof of his identity. So these aren't typical examples of an average benign domain name.

Then, we have also about 17,000 labels of malicious registrations. These are domains where abuse was detected or where it was very clear from the domain name that they would be used for abuse. And again, there are about 5000 labeled as benign, but it is very similar to the good WHOIS labels. So these are registrations where something was suspicious but in the end, nothing malicious was done.

So we have quite a lot of labels here, but just to put things into perspective, still more than 96% of our registrations are unlabeled. It's also interesting to look at the overlap here between the malicious and the bad WHOIS labels. What you see is that over two thirds of all malicious registrations are also bad WHOIS. But that is actually something that changes over the course of the dataset. So in the second [half of our] data set, all bad WHOIS or all malicious registrations are also labeled as bad WHOIS.

Then one issue with these labels is that they are incomplete. Remember, we have this 96% of registrations that are unlabeled, and you can't assume that all of these are actually good WHOIS or benign registrations.

To give some proof of that, I represented here all registrations by a single registrant, and each dot here represents one unique

registration. Dots with the same color use the exact same combination of contact details, so the same name, same phone number, e-mail address, things like that.

And then what you see here is that in the first part in 2019 the registrant verified his contact details quite a bit, but none of these registrations reflect as being bad WHOIS. And then you have here a period in 2020 when the registrant consistently used the same contact details, and all of these were labeled as bad WHOIS except for a few here in October 2020 which were not flagged as bad WHOIS although the same contact details were used.

So I think overall, something like 30% of the 244 registrations by this registrant were not flagged as being bad WHOIS although they probably should have been as they were all done by the same registrant and using very similar contact details.

And I think this one is a bit less obvious, but all the registrations that reflect as bad WHOIS also might be malicious. The problem here is that as soon as a registration is identified as being bad WHOIS it is not added to the zone. So it's impossible to see which website would have been linked to it, and therefore it's impossible to determine whether it would have been used for doing malicious things.

So the fact that the ground truth here is incomplete caused troubles both during training and during evaluation. During training it cause problems because if you give it a lot of very

similar examples and half of these examples have a malicious label and half of them have a benign label, then that will obviously confuse the classifier. And during evaluation these incorrectly labeled examples with masquerade as false positives which will make a classifier look a lot worse than it actually is.

So what initially seemed like a great ground truth here are actually weak labels. At first I considered it would be it would be better to leave out what I first called the benign and good WHOIS labels because there are already examples of suspicious registrations and that would confuse the classifier. But then that means that we are only left with positive examples, and to train a classifier we also need negative examples.

So to get these negative examples we have to make some additional assumptions. The first assumption is that when no incidents are detected for registration 30 days after its registration, then it was probably a benign registration. Second, when a registration uses contact details that are already on the blacklist, then it's probably a bad WHOIS registration. And then third, when a domain name that is registered contains critical keywords like the name of a bank, then it was probably a malicious one.

And then based on these five weak labels, we construct two training labels. The first one is a is_bad_whois label which is based on the original bad WHOIS label and then we make it a

negative example if no incidents were detected 30 days after the registration. Then the second, the needs_attention label, combines all of our weak labels. And this label is true when it was originally labeled as malicious, as bad WHOIS, or when contact details on a blacklist are detected, or when it contains critical keywords. And it is labeled as false when no incidents are detected 30 days after the registration.

So then we can finally move on to the machine learning pipeline. Our machine learning pipeline starts with some preprocessing with [critical] things like computing machine inputting missing values and encoding categorical features.

And then we train the first classifier on the bad WHOIS label. But here we only use the features as input which are based on the registrant's contact details. So we don't use the registration history features and features or reputation scores that we derive from the name servers and the registrars that are used.

And then we train a second classifier on the needs attention label. And as the input we use the prediction as a feature. So we use the prediction of the bad WHOIS classifier as a feature as well as all other training features.

So actually what you have here is a two-step classifier that first tries to determine for a registration whether it contains fake contact details and then in a second instance tries to determine

whether it was made with the intent of doing something malicious.

EBERHARD LISSE: You're running at the end of your time.

PIETER ROBBERECHTS: So I think I need three or four more minutes.

EBERHARD LISSE: Carry on. Carry on. Try to….

PIETER ROBBERECHTS: So then as a baseline we use this expert-based classifier which I mentioned at the beginning of my talk. And here the idea is to assign a point to a registration for each of the rules below that it violates and then classify it malicious as soon as the number of points exceeds a given threshold.

And the choice of threshold is a tradeoff between two metrics: precision and recall. So recall measures how many malicious domains are selected. So if you choose a very low threshold, you will select almost all registrations. And therefore you will also automatically select almost all really malicious registrations. But that will be at the cost of a very low precision because among all

domains that you will have selected there will be a lot of false positives.

So basically, what you have here is a tradeoff between how many of the [inaudible] malicious registrations do you detect at the X axis and the amount of manual work you have to invest in filtering out the false positives, ones on the Y axis.

And then these are the results of the bad WHOIS classifier. So again, the green line here represents the results of the rule-based classifier but now with a confidence interval of one standard deviation.

Then in yellow I've shown the results of a logistic regression classifier. So this is basically the same as the rule-based classifier, but the main difference is that we now give variable rates to the rules. So here you could assign half a point for violating some rule and two points for violating other rules.

And then finally the purple line shows the results of a light gradient boosting model which is sort of the state of the art nowadays to solve these kinds of problems. And you'll see that this one is performing the best.

So if you select a point on this curve, you can see that we can select about 38% of the bad WHOIS domains at a cost of 59% false positives. The good thing is that this is significantly better than

the rule-based classifier. But it's probably still not accurate enough to fully automate registrant verification.

Although, one important remark here that we have to make is that we don't really know how complete the ground truth is. So I think we can be pretty confident about the recall for a given threshold because everything that was flagged as being bad WHOIS was manually verified. But for a given threshold we don't really know how many of the unlabeled examples we missed. So actually, the precision might be a bit higher here than this graph seems to show.

And we can do the same for the needs attention classifier. And maybe the results here seem a bit surprising because now it seems like the light gradient boosting model is performing the worst. But I hope you remember that we trained this on different labels. So during training we learned this classifier that all registrations that included blacklisted contact details should need attention. And then we evaluated here only the bad WHOIS and the malicious labels.

So what has happened here is that this classifier has found a lot of registrations which use blacklisted contact details that were not flagged as being bad WHOIS in the training data. And that's something you can see as well if we evaluate it only on non-blacklisted registrations. So we moved all registrations that contained blacklisted contact details from our test set and then

reevaluate the precision and the recall, and you can see that the light gradient boosting model performs the best again.

So then in the end it still has to be verified by a human. And then it tells if you can explain your predictions. And therefore we use [inaudible] values. So let's go back to the example that I showed in the beginning, the myaccountverify.be example. And indeed, we found that this is a malicious registration. And that is so because it has found that the population of the city that the registrant provided is 15 million and the classifier has learned that malicious registrations often come from these huge [inaudible] cities.

The second reason is because the email address contains only a very small portion of the name of the registrant. And then third, it contains a suspicious keyword which I think here [inaudible] will verify.

EBERHARD LISSE: Really must come to an end.

PIETER ROBBERECHTS: Yes, this is the end.

EBERHARD LISSE: Okay.

PIETER ROBBERECHTS: So maybe three short takeaway messages. We've shown that abusive registrations have distinct properties, and based on these properties we can construct a set of features and train a machine learning algorithm that outperforms a rule-based system. But the tricky thing here is the ground truth which is probably a bit biased toward the rule-based system based on which the ground truth was constructed. And incompleteness of ground truth makes it really hard to evaluate this.

And that's it.

EBERHARD LISSE: Thank you very much. Sorry about this technical issue cutting into your time, so I can't allow any questions. I see [inaudible] making questions in the chat, and I was thinking similar the same thing. I would like to see an open source engine where you can do this in some form of rule-based so that Nominet can plug their things in and you plug theirs and whoever has got other ideas plug it in so that would refine the model to become more specific.

Thank you very much. Good presentation but I need to move forward now to Mats Dufberg who will talk about Zonemaster. Mats, you have the floor. You are still on mute, Mats.

MATS DUFBERG: Okay, thank you. Please, next slide. Or should I present?

EBERHARD LISSE: If you're happy, let the presenter use it. There it is. I don't see it yet, but carry on.

MATS DUFBERG: I see the slides.

EBERHARD LISSE: Carry on, carry on, carry on. Don't worry.

MATS DUFBERG: Please, next slide. Okay, so keeping DNS healthy, everybody knows that it's very crucial for all services on the Internet. And many times if DNS fails or there is a problem in DNS, it could be interpreted as network issue instead. So we should keep track of DNS to make sure that DNS does not create the issues for users.

DNS is, however, quite complex to troubleshoot. And since we've added DNSSEC to it, it makes it even harder to troubleshoot it. So that's why we need a way to check it. Next slide, please.

So Zonemaster is such a tool to check the health of the DNS. It checks the delegation. It verifies name servers so that they give consistent responses. And also, it verifies that all name servers respond to queries. It's quite important. Several times you can

see that a zone is set up. It seems to work very well. But it does not have more than one name server in practice. Zonemaster also verifies DNSSEC which is much harder for the ordinary user to check itself.

Zonemaster is focused on entirely geared at the hosting server, so it does not check resolvers. That's for a different kind of tool. Next slide, please.

So if you go to zonemaster.net and check, you will get a Zonemaster available for you to check your domain. And here if you enter icann.org and click on check—next slide, please—you'll get a report in a couple of minutes or a minute, depending on the size of the zone and the complexity, the number of name servers and if they respond, etc. So it's there, available to anyone to check their domain or check any domain. Next slide, please.

But Zonemaster can be used for other purposes, not only checking your domain using the GUI. You can use it for troubleshooting. You can use it for monitoring. You can use it for statistics and measurement. And you can verify your domain before it's even registered. You set it up in DNS. You check it will it work if I register it or before you delegate it will it work. And you can also do a check of all domains under your TLD which could take a very long time if your TLD is a large one. But if it's a small TLD, it doesn't take a very long time. Next slide, please.

So who is behind this Zonemaster? It's a cooperation between the IES or Internetstiftelsen in Sweden and AFNIC. We together form our project that operates and maintains and develops Zonemaster. It's an open-source tool, so anyone can go and get it. Anyone can install it. And anyone can contribute to the development of Zonemaster. So one type of contribution is, of course, if you see that something is missing in Zonemaster or something is not working as you expect it, you can report it and we will try to fix it. Next slide, please.

So all the tests that Zonemaster does are based on standards, the RFCs and the best practices. And all tests are defined in written specifications, so you can go and check and see what you expect Zonemaster to do. So you don't have to read the code to find out Zonemaster is supposed to do. You can read it in a specification.

It's modular built, so it's possible to integrate parts of it for your tool or your usage. It's actively maintained. So we have releases twice a year. And as someone noted, we had a release just two or three days ago, and the next release is expected in the fall. And it's also translated into several languages. So if you go to Zonemaster.net, you will see that you can select the language that you want the report in. And those are the seven languages that are available. Next slide, please.

So one new feature of Zonemaster is the check of CDS and CDNSKEY. If you attended the DNSSEC workshop this morning,

you would hear a lot about CDS and CDNSKEY. Some ccTLDs, for example .se, check the delegated domain names for CDS and do update on the DS record based on that. Next slide, please.

So if you check your domain name and you have added a CDS or CDNSKEY, Zonemaster will check and verify that it's according to the standards. Next slide, please.

So there could also be policy for the TLD, but that is not possible for Zonemaster to check. So that has to be tested against the TLD. For example, requirements for DS boot strapping. So if you want to read about the specifications or how the test, what it's doing, you read the specifications. And they are in GitHub. Next slide, please.

The last couple of releases we have improved the logic, improved the messages, and we've tried to decrease the messages that Zonemaster outputs to have them more focused. Next slide, please.

So as I said, Zonemaster is translated into several languages, and the latest addition is Spanish. And all translation is done by ccTLDs or people from ccTLDs and volunteer for those. And if you wanted to contribute by adding your language to Zonemaster, you're welcome to contact us and we can add that into the releases of Zonemaster. But you could also install Zonemaster on your own premises and add your language because it's quite

simple. There are instructions how to add another language. Next slide, please.

So zonemaster.net is only one of the installations of Zonemaster. We know that there are many installations of Zonemaster, both installations that present the GUI for the world and private installations that were in batches or integrated into systems. Next slide, please.

So integration is an important feature of Zonemaster and the possibility of doing integration. Using the web GUI is only one way of using Zonemaster. Zonemaster is also available as a CLI tool, and that CLI tool is of course more lightweight and easier to install.

And if you want to do integration of Zonemaster into your system, there are several ways of doing that. You can directly integrate the Zonemaster-Engine Perl libraries into your system. And that maybe requires that you have support for Perl. But there are other ways. One is to wrap around the Zonemaster-CLI tool. And you can integrate that into any language. Or you can integrate into the Zonemaster-Backend which is the way that the GUI works. And then you use RPC-API calls to the backend. And that also includes a database where you store the completed tests that can be retrieved later. Next slide, please.

You can also test Zonemaster on Docker. It's available on Docker Hub, and that will work for any system that supports Docker. So

if you have Docker installed on your laptop, you can just run the command “docker run -t --rm zonemaster/cli --no-ipv6” and test icann.org.

This --no-ipv6 is needed if you sit or preferred if you sit on a no IPv6 environment or if you don’t run Linux on your laptop. That is a limitation of Docker. Docker on Windows and Mac does not support IPv6 even if you have IPv6 on your laptop. There’s a link there that you can go to and see how you use Zonemaster on Docker. And this way you don’t need to install anything. The command will fetch the image and run the test for you. Next slide, please.

As I said, results are stored in a database if you run tests against Zonemaster-Backend. That’s the way that the GUI does. But you can also use the RCP-API, as I mentioned. And if you run a test, you can create a link from the GUI and you can send that to someone who can retrieve the same test again from the database. Next slide, please.

A thing that we have worked on is to improve the batch function, and that especially in the new release, so the load of a batch run is much lower. So the batch function is part of Zonemaster-Backend. So you can run several domain names at the same time. For example, all parts of the domains of your TLD. And then after the batch has been completed, you can create URLs to the GUI that you attach to the same backend. And those could then be

sent to the registrants or the registrars or however you want to make it available. Next slide, please.

So as I said, Zonemaster is an open-source project. You get full documentation, full source code on GitHub. The license is permissive, so you can integrate it in your system and use it for your use. Zonemaster.net is a reference installation, and zonemaster.iis.se is running against the same backend but with a different GUI.

Thank you. The next slide is thank you and questions.

EBERHARD LISSE: Thank you very much. I will take one question if there is one from the floor because we are running into the break even though they are not cleaning the rooms in between so I'm not too hung up on it. I don't see anything. I don't see anything in the chat. And I saw just a remark from Jacques on the previous presentation, we should have a GitHub project and somehow in-community crowdsource a list classifiers a la zonemaster. So you're influencing other projects as well.

Thank you very much for doing this on a relatively short notice and before the break. I will then give us all a break for about 22 minutes.

MATS DUFBERG: Yeah. Questions can also be sent to my e-mail address.

EBERHARD LISSE: Yes, the email address of every presenter is in the agenda, clickable. Thank you.

[END OF TRANSCRIPTION]