DNS Protocol Parameters Analysis

Abstract

We propose a methodology for continuous measurement of several "Identifier Technology Health Indicators" metrics: overhead in root traffic, leakage of undelegated names at the root, and usage of parameters defined in DNS registries. The proposed methodology is to:

1) *Use multiple collection points*
2) *At each collection point, perform 15min of collection at random hour, collected every day.*
3) *Data from the various collection points is aggregated to compute the metrics.*

The collection is performed by running a collection tool provided by ICANN. The tool removes any PII information from the observed data, and produces a summary table. The summary tables are collected every day, and are processed by an aggregation tool to provide the required metrics.

Monitoring DNS Protocol Parameter Usage at Resolvers

ICANN is developing the monitoring of "health metrics" for various Internet Identifiers, using a set of "Identifier Technology Health Indicators" (ITHI). The identifiers include Names, Numbers, and Protocol Parameters. A general presentation of the ITHI work is available here: https://www.icann.org/ithi. The analysis of Registered DNS Protocol Parameters focused on a subset of the ITHI problem, specifically the computation of the IHTI metrics M3, M4 and M6:

- overhead in root traffic (M3.1, and M3.2 for duplicate requests contained in single PCAP file)
- leakage of RFC6761 names and other undelegated names (M4)
- usage of DNS protocol parameters defined in IANA registries (M6) plus list of unregistered parameters and frequencies.

Preliminary studies showed that the metrics M3.1, M3.2 and M4 can be obtained by analyzing captures of DNS root servers traffic, but that the metric M6 cannot because the root traffic only provide a biased view of DNS parameter usage. Indeed, if resolvers implemented DNS Query Name Minimization to Improve Privacy (https://datatracker.ietf.org/doc/rfc7816/), the root traffic would contain almost no information about DNS Protocol Parameters usage. In contrast, DNS resolvers are in an excellent position to monitor this usage.

Collecting data about DNS usage at resolvers poses issues of privacy and operational overhead, which we believe are addressed in our proposed design.

Definition of metrics

M3 metrics for overhead in root traffic

The root traffic overhead is measured with 2 metrics:

- M3.1: % of NX domain response over the total number of responses by root servers

- M3.2: % of queries that should never have been sent, because the sending resolver already received an NS record for the specified TLD, and the TTL of that NS record has not expired yet.

M4 metrics for name leakage

The name leakage is measured by the M4 metrics. M4 encompass a list of "Top-N" strings seen at the root that have not been delegated by ICANN or put on the RFC6761 "Special Use Names".

> ***Question: Should we also attempt to list the random generated names? Obviously we cannot post all of them, but there may be value in isolating specific patterns and associating them with particular software components.***

M6 metrics for DNS Protocol Parameters Usage

We plan to compute the M6 metric for three groups of registries managed by IANA: the Domain Name System (DNS) Parameters, the Domain Name System Security (DNSSEC) Algorithm Numbers, and the DNS-Based Authentication of Named Entities (DANE) Parameters. These three groups include the following set of parameters:

Group	Parameters	Metric Index
DNS	DNS CLASSes	M6.DNS.01
	Resource Record (RR) TYPEs	M6.DNS.02
	DNS OpCodes	M6.DNS.03
	DNS RCODEs	M6.DNS.04
	AFSDB RR Subtype	M6.DNS.05
	DHCID RR Identifier Type Codes	M6.DNS.06
	DNS Label Types	M6.DNS.07
	DNS EDNS0 Option Codes (OPT)	M6.DNS.08
	DNS Header Flags	M6.DNS.09
	EDNS Header Flags (16 bits)	M6.DNS.10
	EDNS version Number (8 bits)	M6.DNS.11
	Child Synchronization (CSYNC) Flags	M6.DNS.12
DNSSEC	DNS Security Algorithm Numbers	M6.DNSSEC.1
	DNS KEY Record Diffie-Hellman Prime Lengths	M6.DNSSEC.2
	DNS KEY Record Diffie-Hellman Well-Known Prime/Generator Pairs	M6.DNSSEC.3
DANE	TLSA Certificate Usages	M6.DANE.1
	TLSA Selectors	M6.DANE.2
	TLSA Matching Types	M6.DANE.3

Each of this set of parameters is associated with a metric index. For each of these indices, we will compute two metrics:

Metric Number	Metric name	Metric definition
M6.X.N.1	Parameter usage for table of metric N in group X	Number of parameters found at least once in real traffic, divided by the total number of parameters found registered in the table.

M6.N.2	Squat rate for table of index N	Total number of instances of unregistered parameters found in real traffic, divided by the total number of parameter instances found in real traffic.

Using these definitions, the M6 metrics would look like this:

Group	Parameters	Metric Index	M6.X.N.1	M6.X.N.2
DNS	DNS CLASSes	M6.DNS.01	80%	0.00%
	Resource Record (RR) TYPEs	M6.DNS.02	43%	0.00%
	DNS OpCodes	M6.DNS.03	100%	0.02%
	DNS RCODEs	M6.DNS.04	61%	0.00%
	AFSDB RR Subtype	M6.DNS.05	0%	0
	DHCID RR Identifier Type Codes	M6.DNS.06	0%	0
	DNS Label Types	M6.DNS.07	75%	0.00%
	DNS EDNS0 Option Codes (OPT)	M6.DNS.08	33%	0.17%
	DNS Header Flags	M6.DNS.09	50%	0
	EDNS Header Flags (16 bits)	M6.DNS.10	100%	0
	EDNS version Number (8 bits)	M6.DNS.11	100%	0
	Child Synchronization (CSYNC) Flags	M6.DNS.12	0%	0
DNSSEC	DNS Security Algorithm Numbers	M6.DNSSEC.1	40%	0
	DNS KEY Record Diffie-Hellman Prime Lengths	M6.DNSSEC.2	0%	0
	DNS KEY Record Diffie-Hellman Well-Known Prime/Generator Pairs	M6.DNSSEC.3	0%	0
DANE	TLSA Certificate Usages	M6.DANE.1	0%	0
	TLSA Selectors	M6.DANE.2	0%	0
	TLSA Matching Types	M6.DANE.3	0%	0

The detailed usage of each parameter is available in the summary files.

Proposed methodology

The proposed metrics could be computed in many ways, but we propose two big guidelines. First, we will rely on passive monitoring, rather than active queries. Second, we will follow a statistical sampling approach, rather than trying to gather all the traffic all the time from all potential sources. The proposed methodology is thus:

4) Use multiple collection points
5) At each collection point, perform 15min of collection at random hour, collected every day.
6) Data from the various collection points is aggregated to compute the metrics.

Justification of proposed methodology

Passive monitoring has the advantage to avoid the "sampling bias" inherent with active queries. For example, suppose the tool would instead actively query the DNS servers of the Alexa Top N web sites. This would certainly return a lot of data, but there would be a bias towards web traffic from big sites, and the DNS data related to for example SMTP servers would be underrepresented. In contrast, passive monitoring will return a fair count of all the name and parameters encountered in regular operation of the DNS.

The metrics that we have defined can be readily approached by statistical sampling. The measurements ultimately depend on the DNS query generation process of a large number of agents. We can expect that process to be somewhat dependent on time and location – previous measurements showed for example the % of NX response at the root varied between 59.1% and 66.7%. However, for a specific time period and location, the numbers converge very quickly and do not vary all that much once we have analyzed a sufficient number of transactions.

The biggest constrain that we have is the analysis of rare events, such as parameter values that are only present in just 0.001% of transactions. As a rule of thumb, we want to analyze at least 1 million transactions to make sure that such events are detected. Analyzing 2 or 5 more million transactions would help us characterize the frequency with a greater precision, as in "between 0.0005% and 0.0015%" instead of just "less than 0.002%", but it would not provide us with a dramatically different insight.

Rather than spending resource analyzing large set of data collected at about the same time and location, we should increase the quality by analyzing several sets of data collected at various times and locations, aiming to analyze on average 1 million transactions at each time and location. The stated 15 minutes collection time will provide for 1 million transaction if the monitored location experiences more than 1100 transaction per seconds, and fewer than 10 million transactions if there are fewer than 11000 transactions per seconds. If a location observed many more than 10000 tps, it would be appropriate to reduce the collection time.

Methodology for the M3 metrics

The M3.1 metric is computed by monitoring a number of transaction to the root servers, and counting the number of responses and the occurences of the response code "normal" and "NX". M3.1 is computed as the ration of number of NX responses over number of normal + NX response. The other error codes such as "format error" or "refused" will be ignored for the purpose of computing this metric.

The M3.2 metric requires checking queries to the root that are duplicate of other queries. When monitoring queries, we will keep a cache keyed by resolver and TLD. When considering a particular query, we will check whether the <resolver,TLD> pair is already in the cache, and if the TTL of that item has not elapsed. If there is a valid cached value, we count the query as invalid, else we count it as valid and enter it in the cache. There is a potential issue happening when the cache is full, in which case we will ignore queries that cannot be cached. The M3.2 metric will be computed as the number of invalid queries divided by the number of valid and invalid queries, ignoring the queries that could not be cached.

Methodology for the M4 metrics

M4 metrics characterize leakage at the root. We will compute three metrics:

- Leakage of RFC 6761 special use domains, providing for each special use domain the % of root queries directed at that domain.
- Leakage of "frequently used" invalid TLD, providing for each of the most used such domains the % of root queries directed at that domain.
- Leakage of "special pattern" invalid TLD, estimated for now by measuring the frequency of use of patterns of specific length.

The "frequently used" invalid TLD will be retrieved by analyzing traces, and retaining the names that appear most frequently. This is done by maintaining a cache of the 32000 last seen entries, a size that pretty much guarantees that domains present in more than 0.1% of queries will all be captured.

Methodology for the M6 metrics

The M6 metrics require tabulating the number of occurrence of specific entries in the protocol parameter tables. When monitoring transactions, each transaction will be scanned for potential presence of registered parameters, keeping a running total of all number of occurrence of each parameter.

For a given table N, the metric M6.N.1 will be obtained by counting the number of registered parameters present in the table. The metric M6.N.2 will be obtained by counting the number of occurrence non-registered parameters (squatting) and dividing it by the number of occurrence of all parameters for the table.

Aggregation of data from multiple locations

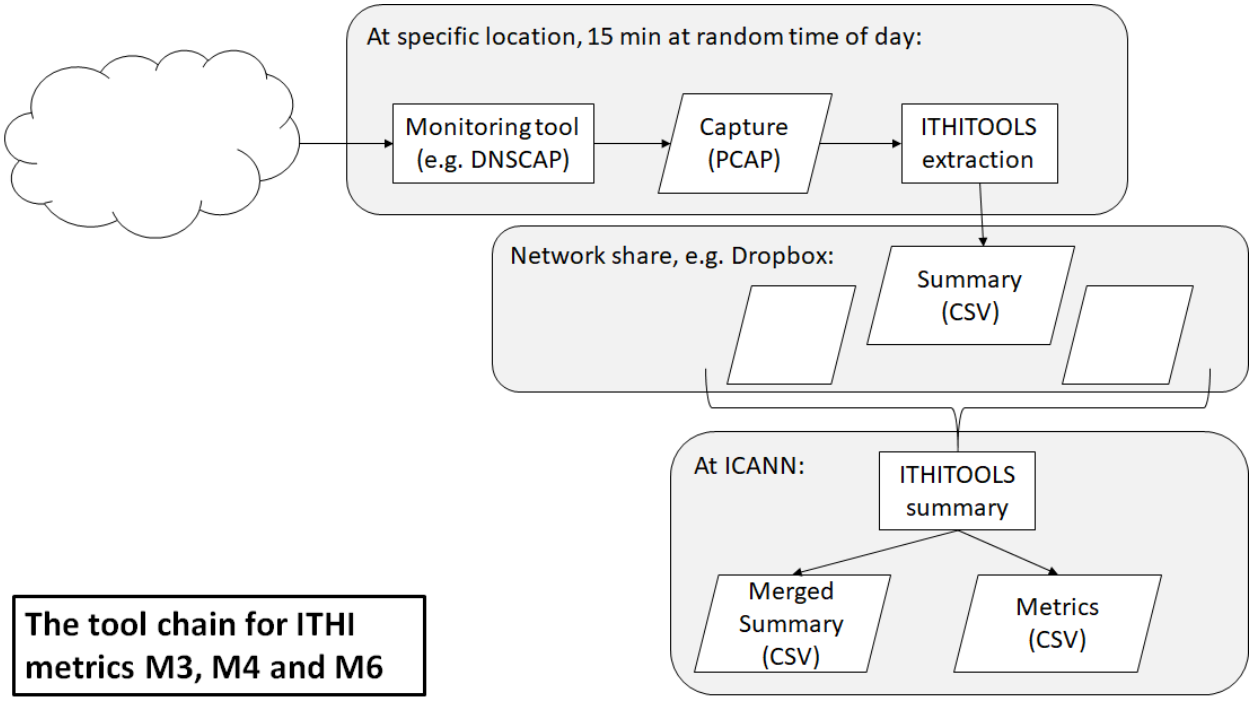
To compute the metrics, we will have to aggregate data from multiple location. The aggregation rules are defined in the following table:

Metric	**Data at time and location**	**Aggregation**
M3.1	Number of NX domain responses and number of No Error responses at location.	Sum of numbers of NX domain response at all locations, divided by sum of total numbers of responses at all locations.
M3.2	Number of useless and useful root queries at location.	Total number of useless root queries at all locations, divided by total number of queries at all locations.
M4.1	Number of root queries directed at each special use domains (per RFC 6761)	Total number of queries directed at each special use domain, divided by total number of queries at all locations.
M4.2	For the top N string present in invalid root queries, the total number of instances is collected. By default, N is set to 128.	For each string presented as top N at any location, compute the total number of instances at all locations divided by total number of queries at all locations. Then sort and extract the present in more than 0.1% of queries.

M4.3	Compute the number of strings of length X in invalid queries for each value of X between 1 and 64.	Compute the totals across all location, divided by total number of queries at all locations. Sort, and present the patterns present in more than 0.1% of queries.
M6.X.N.1	Count the number of instances of each registered parameter.	Sum the total number of instances of each registered parameter at all locations. Then count the number of parameters present in the table.
M6.X.N.2	Count the number of occurrence of non-registered parameters. Count the total number of occurrences of parameters in the table.	Sum the count of occurences of non-registered parameters for all locations, and divide by the sum of occurences of

Tool structure

The metric computation is performed by the DNS Analysis tool (ITHITOOLS) developed for ICANN (https://github.com/private-octopus/ithitools). The tool supports two modes of operation: collection and summarization. The principle of operation are described in the following diagram:



The tool chain for ITHI metrics M3, M4 and M6

The capture starts at a specific capture point. A tool like DNSCAP or Wireshark is used to capture 15 minutes of traffic at a random time, each day. Once the capture is complete, the corresponding PCAP file is analyzed by the ITHITOOLS tool which creates a summary file, encoded in CSV format and saved at on a specified network share. Each day, the ITHITOOLS tool is run by IANA in summary mode. It collects the summaries added the previous day to the network share, and aggregates them to produce the required ITHI metrics.

Tool parameters

The ITHITOOLS tool can be configured with a set of parameters, described in the following table:

Parameter	Explanation
-p file	PCAP file to analyze. This parameter instructs ITHITOOLS to run in “extraction” mode.
-s file [..[file]]	List of summary files that should be aggregated. This parameter instructs ITHITOOLS to run in “summary” mode.
-o file	Output file where to place the computed summary.
-m file	Output file where to place the computed metric.
-r root-address-file	List of IP addresses used by root server, in text format, one address per line. Used to differentiate traffic to or from the DNS root from traffic to other DNS resolver. Only traffic to or from the DNS root is considered for metrics M3 and M4. By default, ITHITOOLS uses an internal static list of source addresses.
-a resolver-address-file	Allowed list of resolver addresses. Traffic to or from addresses in this list will not be filtered out by the excessive traffic filtering mechanism. By default, ITHITOOLS does not privilege any resolver addresses.
-x resolver-address-file	Excludes list of resolver addresses. Traffic to or from these addresses will be ignored when extracting traffic. By default, ITHITOOLS does not exclude any resolver address.
-f	Filter out address sources that generate too much traffic. This is meant to exclude traffic from for example test tools, whose atypical patterns could skew the statistics. By default, ITHITOOLS does not perform excessive source filtering.
-d table-name:csv-file	Use the corresponding definition for the specific parameter table “table name”. The CSV file should be downloaded from the IANA site, using the “CSV” link provided by IANA for the table. By default, ITHITOOLS uses precompiled values of the parameter tables. However, new values may have been registered between the last ITHITOOLS update time and the time at which the metrics are computed. Specifying an up-to-date table ensures correct computation of the “squat” metrics (M6.N.2). There is no need to specify this files during simple extraction operations.
-n number	Number of strings that should be returned as part of the list of leaking domains (M4). By default ITHITOOLS will include in summaries the top 128 leaking names.
-u tld-file	Text file containing the list of TLD reserved via the RFC6761 process. By default ITHITOOLS used an internal static list of reserved TLD. However, as new special use TLD get authorized by IETF, that list can evolve, which would lead to erroneous computation of the M4 metric. There is no need to specify this file during simple extraction operations.

Data present in the summary file

The summary file is organized as a large table, encoded in CSV format. The table has the following columns:

- Table friendly name,

- Entry type (0 for numeric, 1 for text)
- Entry value
- Parameter friendly name,
- Instance count.

The tables carry the input to the metric computation. They are designed to only require minimal parameterization at the collection site, and enable precise computation during aggregation.

The tables used in the various metrics are:

Metric	Table	Entries	Count
M3.1	NX responses	NX (3)	Number of NX responses
		No Error (0)	Number of valid responses
M3.2	Valid queries	Valid (1)	Number of valid queries
		Invalid (0)	Number of invalid queries
M4.1	RFC 6761 TLD	Special Use TLD value	Number of occurences of this TLD string in queries
M4.2	Leaked TLD	TLD Value	Number of occurrence of the leaked TLD in queries. Only computed for the “most frequent” leaked TLD.
M4.3	Leaked by length	TLD Length	Number of occurences of TLD with this length
M6	Table name, per IANA	Parameter value defined in the table. Friendly name is documented if present, but is really optional.	Number of occurences of this specific parameter value in responses

Privacy Considerations

The ITHICAP tool chain is designed to minimize risks of leaking PII information. We recognize that information present in the capture file includes addresses and names that are sensitive from a privacy point of view. We exercise the following mitigations:

1) The ITHITOOLS tool performs extraction of data without requiring any network access. It does require that the extracted summary be copied on a network share, but this can be performed off-line, and the copying to the network share can be performed later using a tool chosen by the operator.
2) The ITHITOOLS summaries only contained counts and statistics.
3) The ITHITOOLS tool is open source, and the code can be readily inspected.