DNS Protocol Parameters Analysis

Christian Huitema (Private Octopus Inc.), Alain Durand (ICANN)
November 27, 2017

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:

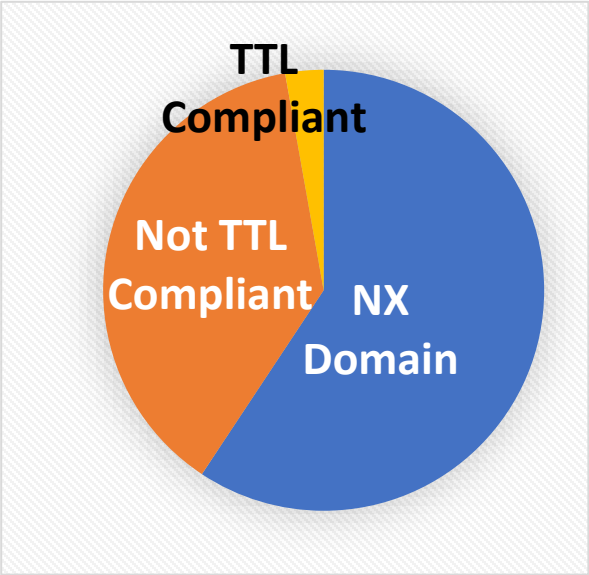
- overhead in root traffic (M3.1, M3.2 for duplicate requests contained in single PCAP file, and M3.3 for analysis of the names causing NX domain responses)
- usage of TLD, and leakage of RFC6761 names and other undelegated strings (M4)
- usage of DNS protocol parameters defined in IANA registries (M6) plus list of unregistered parameters and frequencies.

Preliminary studies showed that the metrics M3.1, M3.2 and M3.3 can be obtained by analyzing captures of DNS root servers traffic, but that the metrics M4 and M6 cannot because the root traffic only provide a biased view of the traffic and 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. If they implemented aggressive caching of the root zone (NSEC3 aggressive), then the root would see almost no leakage of names and would be unable 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



Previous observations have shown that the traffic encountered at the root includes a large fraction of overhead. In previous measurements, we saw about 60% of queries generating an NX domain responses. Even among the queries that generate no error, we found that many queries would not have been needed if the resolver was "TTL Compliant", i.e. if it had cached previously received responses for the duration of the TTL. We want to track this overhead at the root, to see how it will evolve over time. This leads to the definition the metrics M3, which are in fact 3 metrics:

- M3.1: % of NX domain response over the total number of responses by root servers
- M3.2: % of queries that should not have been sent if the sending was "TTL Compliant".
- M3.3: Investigation of the "name leakage" at the root that causes the NX Domain responses measured in M3.1.

M3.3 metrics for name leakage at the root

The name leakage at the root is measured by the M3.3 metrics. Earlier measurements showed that these leakage corresponds to several classes of name. We want to measure the evolution of this leakage over time with M3.3, which is composed of 4 metrics, M3.3.1 to M3.3.4.

The first class of leaks corresponds to special use domain names such as for example ".LOCAL", reserved by the IETF according to RFC 6761, and recorded by IANA in the "Special Use Domain Names" registry (https://www.iana.org/assignments/special-use-domain-names/special-use-domain-names.xhtml). For each of the special use domain name, the tools will compute:

- M3.3.1.<TLD>: % of root queries directed to that special use TLD.

The second class of leaks corresponds to commonly configured "unregistered" strings, such as for example ".HOME". These strings are typically configured in private networks, which their administrators believed to be isolated from the Internet. However, experience shows that such isolation is imperfect. The tools will find out which of these unregistered strings appear most frequently in the traces, and will compute:

- M3.3.2.<string>: % of root queries directed to that unregistered string.

We also observe that a large fraction of the leaks is caused by "made up" names. Some of these appear to be the output of Domain Generation Algorithms (DGA). Some are generated by software trying to assess local connectivity to the DNS by sending a query to the root server; doing so with a newly generated DGA name increases the probability that the query will be sent all the way to the root. Other

may be the result of test tools, attack tools, configuration errors, or plain software bugs. The tools will attempt to characterize use of such patterns by computing:

- M3.3.3.<pattern>: % of root queries directed to the specified pattern.

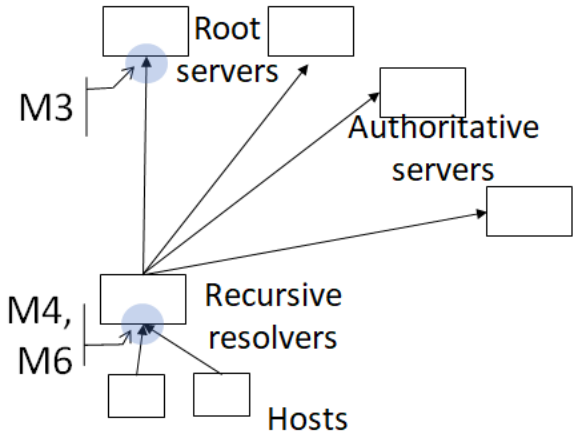
Finally, we will also measure the fraction of leaks that are not explained by M3.3.1, M3.3.2 or M3.3.3:

- Total NX Domains = M3.3.1 + M3.3.2 + M3.3.3 + **M3.3.4**

This definition assumes that there is no double-counting, and that for example a given string cannot be accounted both as an individual string in M3.3.2, and as part of a pattern in M3.3.3.

M4 metrics for name usage and leakage

The M4 metrics measure the usage of TLDs and leakage of undelegated strings by DNS users. As shown on the figure on the right of this text, this is not the same measure as M3, because most users do not contact the root servers directly. Instead, they will contact a DNS Recursive Resolver, which may be provided by their connectivity provider, by their organization, or by a third party. These recursive resolvers will often respond to queries with cached responses, so the root will only observe a very small fraction of the queries. The goal of M4 is to measure the users' behavior, which will be done by analyzing traffic at recursive resolvers.

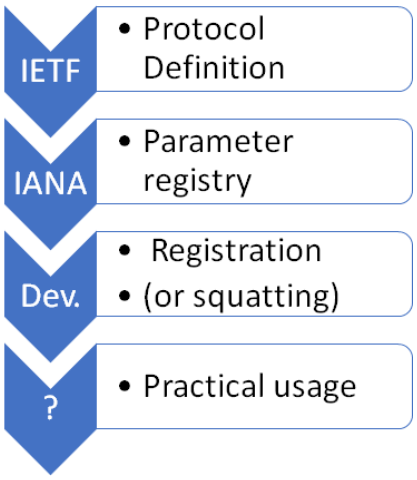


The usage of TLD and leakage of undelegated names is measured in M4 by 4 metrics:

- M4.1: Usage volume of delegated TLDs
    - Fraction of queries directed at delegated TLDs.
- M4.2: Leakage of RFC 6761 Special Use Names
    - For each RFC 6761 name, fraction of queries directed at <name>
- M4.3: Leakage of frequent non delegated strings
    - Find most frequent non delegated top level strings in queries
    - Retain name if fraction > 0.1%, List < string>, fraction of query
- M4.4: Leakage of other strings
    - All queries at non registered strings not in M4.2, M4.3

M6 metrics for DNS Protocol Parameters Usage

The IANA manages a set of parameter registries on behalf of the IETF. These registries are requested by the IETF, as part of a protocol definition. The IANA will then establish the corresponding registry. Developers are then supposed to register new values when the need arises. For example, developers who find the need for a new DNS Record Type will document that new type in an Internet Draft, and get the new type registered by IANA in the DNS RR Type registry. Of course, there is a risk that some developers will find that process too cumbersome, and will instead use an unregistered value, a process that we call "squatting". The goal of the M6 metrics is to measure if registered values are used, and whether we observe squatting. We want to tract this squatting and usage over time, and we also want to track the usage level of specific parameters.



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:

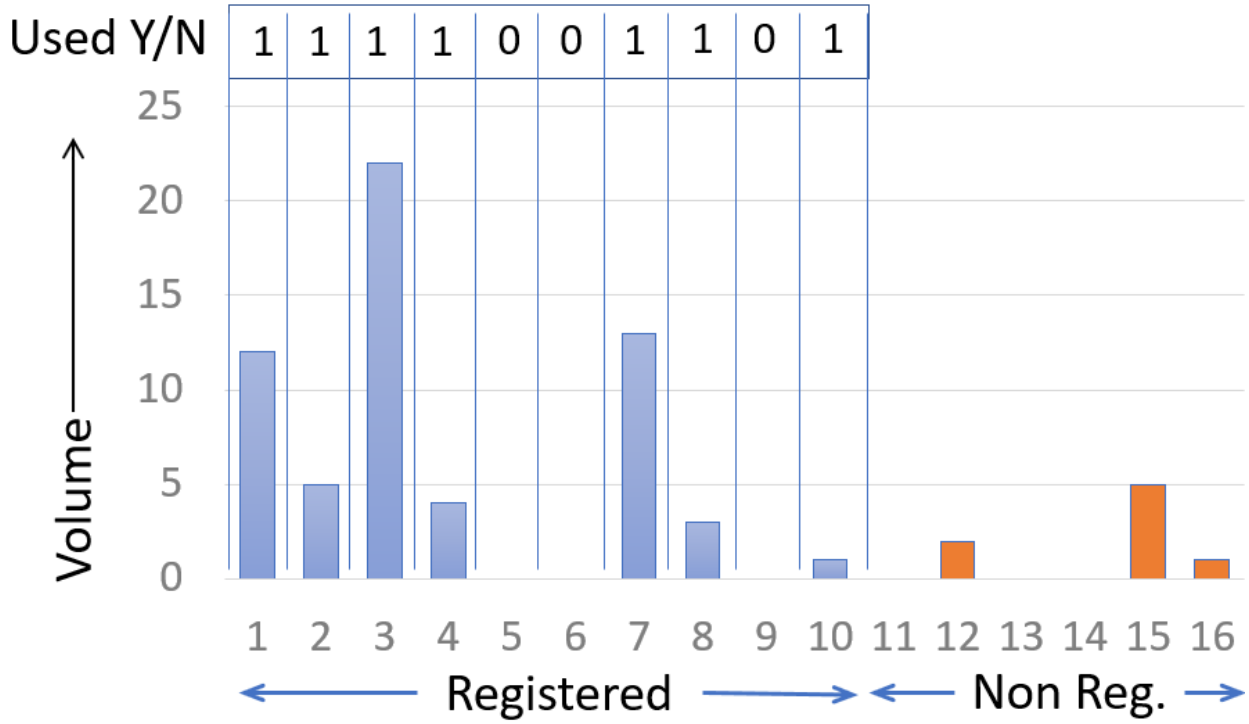
Group	Parameters	Metric Index
DNS	DNS CLASSes	M6.DNS.1
	Resource Record (RR) TYPEs	M6.DNS.2
	DNS OpCodes	M6.DNS.3
	DNS RCODEs	M6.DNS.4
	AFSDB RR Subtype	M6.DNS.5
	DHCID RR Identifier Type Codes	M6.DNS.6
	DNS Label Types	M6.DNS.7
	DNS EDNS0 Option Codes (OPT)	M6.DNS.8
	DNS Header Flags	M6.DNS.9
	EDNS 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 three 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.X.N.2	Squat rate for table of index N in group X	Total number of instances of unregistered parameters found in real traffic, divided by the total number of parameter instances found in real traffic.
M6.X.N.3.<P>	Usage level of registered parameter P in table of index N in group X	Number of instances of the registered parameter in real traffic.

The computation of these two metrics can be explained by using as example fictitious registry that could manage a total of 16 values. In our example, values 0 to 10 have been properly registered. Values 12, 15 and 16 are not registered, but they do appear in some the traces.



To compute the "usage" metric, we look at the traffic observed for the registered values. We see some traffic for values 1, 2, 3, 4, 7, 8 and 10, and mark a "1" in the corresponding "Used Y/N" table; there is no traffic for the values 5, 6, and 9, so we mark a "0" in the table. If we sum the ones, we get 7 registered parameters out of 10. The usage metric is thus:

- Usage = <number used> / <number registered> = 7/10 = 70%.

To compute the "squatting" metric, we compute the total volume of traffic for the squatted values 12, 15 and 16, which in your example is 8 units. The total traffic for all the registered value in our example is 60 units. The squatting metric is thus:

- Squatting = <volume non registered>/<volume total> = 8 / (60 + 8) = 11.8%

Using these definitions, and using numbers computed in a previous study, the M6 usage and squatting metrics would look like this:

Group	Parameters	Metric Index	M6.X.N.1	M6.X.N.2
DNS	DNS CLASSes	M6.DNS.1	80%	0.00%
	Resource Record (RR) TYPEs	M6.DNS.2	43%	0.00%
	DNS OpCodes	M6.DNS.3	100%	0.02%
	DNS RCODEs	M6.DNS.4	61%	0.00%
	AFSDB RR Subtype	M6.DNS.5	0%	0
	DHCID RR Identifier Type Codes	M6.DNS.6	0%	0
	DNS Label Types	M6.DNS.7	75%	0.00%
	DNS EDNS0 Option Codes (OPT)	M6.DNS.8	33%	0.17%
	DNS Header Flags	M6.DNS.9	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

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.

M3.3 metrics characterize leakage at the root. We will compute four 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" non-registered strings, providing for each of the most used such string the % of root queries directed at that string.
- Leakage of "special pattern" invalid strings, estimated for now by measuring the frequency of use of patterns of specific length.
- Amount of leaks not explained by the three metrics above.

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.

The “special pattern” analysis is still a work in progress. In the initial version of the tool, we will consider only one pattern characteristic, the length of the TLD.

Methodology for the M4 metrics

The usage of TLD and leakage of undelegated names is measured in M4 by 4 metrics.

Metric M4.1 measures the usage volume to delegated TLDs. The extraction tool will be parameterized with the current list of delegated TLDs, which is available from the ICANN servers at http://data.iana.org/TLD/tlds-alpha-by-domain.txt. When tabulating queries, we will extract the top level string from the queried name. If it is present in the registered list, we will add one to the count of instances for delegated TLDs.

If the top level string is not present in the registered list, we will check whether it correspond to one of special use names reserved according to RFC 6761. If that is the case, we will count an instance for the corresponding special use name, in order to compute M4.2.

The strings that are neither delegated nor reserved according to RFC 6761 are candidate for the “frequently used” strings tabulated in M4.3. The metric will retain 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 strings present in more than 0.1% of queries will all be captured.

The 0.1% limit cannot be enforced in at an individual capture location. Instead, at each location, the extraction tool will tabulate the N most frequently used strings, with N being set by default to 128. The summarization tool will then merge these results, and extract the most frequently used strings across all locations.

The metric M4.4 will be computed as the complement of the sum of M4.1, M4.2 and M4.3 across all strings.

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:

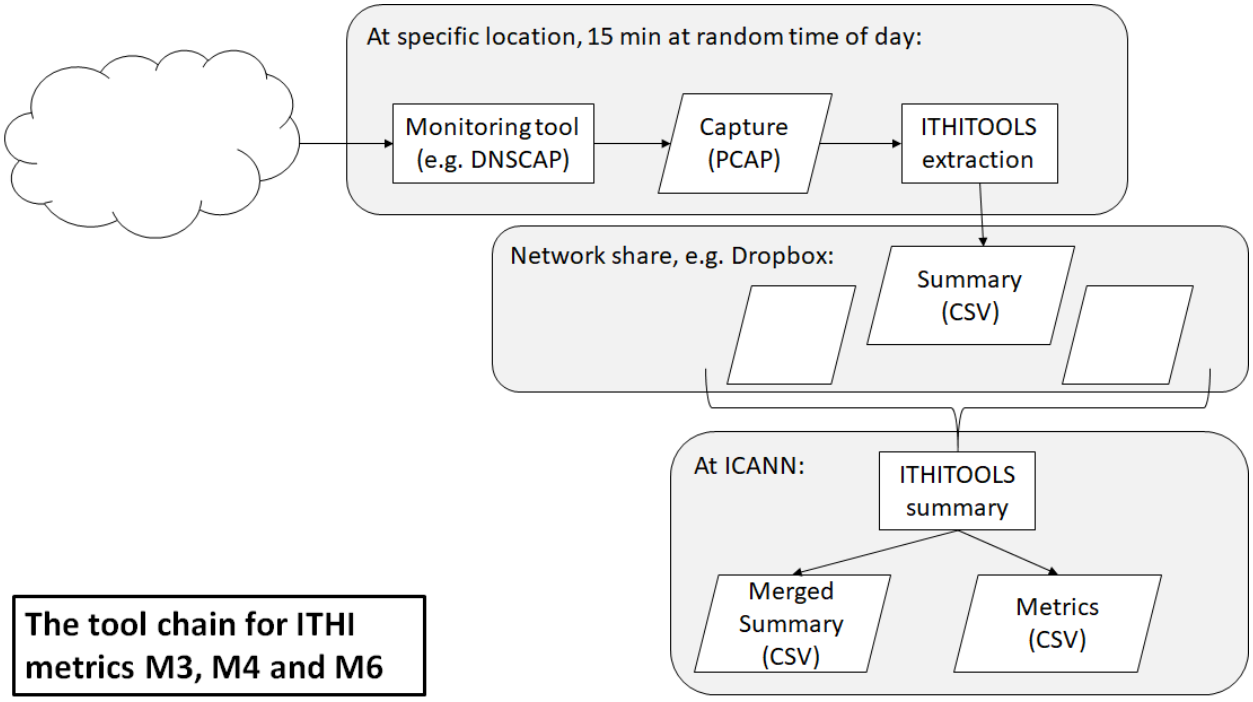
Metric	Data at time and location	Aggregation

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.
M3.3.1	Number of root queries directed at each special use domains (per RFC 6761)	Total number of root queries directed at each special use domain, divided by total number of queries at all locations.
M3.3.2	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 root query 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.
M3.3.3	Compute the number of strings matching specific patterns	Compute the totals across all location, divided by total number of root queries at all locations. Sort, and present the patterns present in more than 0.1% of queries.
M3.3.4	Compute the number of NX Domain root queries not explained by M3.3.1, M3.3.2, and M3.3.3	Compute the remainder after computing M3.3.1, M3.3.2 and M3.3.3.
M4.1	Number of user queries directed at each registered TLD	Total number of queries directed at registered TLDs, divided by total number of queries at all locations. This excludes queries directed at root servers.
M4.2	Number of user 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. This excludes queries directed at root servers.
M4.3	For the top non registered N top level string present in user 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. This excludes queries directed at root servers.
M4.4	Number of user queries directed at other strings than those accounted for in M4.1, M4.2, and M4.3	Compute the remainder after computing M4.1, M4.2 and M4.3.
M6.X.N.1	Count the number of instances of each registered parameter.	Sum the total number of instances of all 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	Sum the count of occurences of non-registered parameters for all locations, and divide by the sum of occurences of

	total number of occurrences of parameters in the table.	
M6.X.N.3.<P>	Count the number of instances of the registered parameter <P>.	Sum the total number of instances of the registered parameter at all locations.

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
M3.3.1	RFC 6761 Special Use Names	Special Use Name value	Number of occurences of this name in queries
M3.3.2	Leaked TLD	TLD Value	Number of occurrence of the leaked TLD in queries. Only computed for the “most frequent” leaked TLD.
M3.3.3	Leaked by pattern	Pattern name	Number of occurences of TLD with this pattern
M4.1	Queries to delegated TLDs		Number of queries to delegated TLDs
M4.1	RFC 6761 Special Use Names	Special Use TLD value	Number of occurences of this name in queries
M4.2	Frequently used non registered strings	TLD Value	Number of occurrences of the leaked string in queries. Only computed for the “most frequent” leaked TLD.
M6	Table name, per IANA	Parameter value defined in the table.	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.
4) The ITHITOOLS code is reviewed by a third party.