Privacy analysis of Tor's in-memory statistics

Karin Herm
The Tor Project
iwakeh⋈torproject·org

Tor Tech Report 2017-04-001
April 2017

Contents

1 Introduction 2

2 Background 2

3 Possible privacy issues 10

4 Mitigate privacy impact 12

5 Impact of implementation changes 18

6 Summary 21

A Appendix 23

Abstract

This report analyzes which possibly sensitive, potentially personally identifying data is stored in memory of Tor relays and bridges or reported to the directory authorities and makes suggestions to reduce the collection and temporary storage of such data.

This research was supported in part by NSF grants CNS-1111539, CNS-1314637, CNS-1526306, CNS-1619454, and CNS-1640548.

1 Introduction

Tor network metrics and the underlying data have been available for many years by now and proven to be a valuable source for analyzing and improving the network as well as for censorship detection.[1]

Tor Metrics' data collecting and processing chain handles various types of data ranging from raw data as measured by running Tor servers[2] to preprocessed and aggregated data ready for further statistical work and as a basis for visualizations.

This report aims at improving privacy protection *before* any data is reported. Of primary interest is the identification of possibly harmful data that is not a necessary part of the running Tor server, e.g. data held in-memory or written to files for providing network/router metrics reports or data written to logs for informative purposes.

Section 2 provides an overview of the Tor Metrics system, its privacy goals, and a more detailed explanation of the metrics collection process as well the associated data. In-memory data with possibly negative impact to client privacy is identified in section 3. Section 4 surveys several measures to reduce privacy impact. Building hereon section 5 details the changes necessary for an implementation of the favored solution. The suggestions made in this report for reducing privacy impact go beyond the scope of a single project and some will need further work to reach the implementation stage. The summary in section 6 takes account of this and also sketches possible next steps.

2 Background

This report assumes the reader to be familiar with the Tor software and Tor network and to some extent with the functionality and data offered by Tor Metrics.

The following sections first present an overview of the data processing chain in Tor Metrics and summarize the privacy goals behind Tor Metrics data collection. Sub-section 2.3 provides a description of the processes of measuring and counting implemented in Tor servers for metrics purposes and shows where in the code the processing of the data takes place.

2.1 Tor Metrics

Tor servers running as relays or bridges publish their presence and capabilities to the directory authorities in form of simple files, the descriptors.

The current system of collecting data about the Tor network is built on descriptors, which are mainly produced and distributed for the operation of the network except for extra-info descriptors, which also provide metrics about the network. This way of measuring the network

[1] Network analysis estimation of cell traffic, estimation of onion services induced traffic and user count estimation as well as censorship detection [9, 10, 8] and all data and visualizations on MetricsWeb.

[2] Here and in the following *Tor server* refers to relays and bridges and other parts of the Tor network fulfilling a server role. The term *client* is used for Tor instances simply connecting to the network. Tor servers report different statistics depending on their configuration. A bridge, a normal relay, an entry guard relay, etc., they all have access to different data and report different statistics.

generates minimal overhead for the network's operation and the data produced is freely available[3] to anyone who cares to collect it. The descriptors are machine and human readable and the knowledge required to make use of them is published in Tor's specification [1].

All descriptors available at a fixed point in time give a good picture of the current status of the network. In order to collect these *pictures* and combine them to a *history* Tor Metrics introduced CollecTor,[4] which gathers and archives the *raw facts* in form of descriptors[5] about the Tor network.

A descriptor document only carries information about a certain point in time, more exactly a time interval, as for example, a measurement interval for extra-info descriptors or the consensus, which applies to the entire network for its valid time interval. Tor Metrics also provides machine and human centered services that create aggregated and enriched data from the descriptor collection. The central services by Tor Metrics building on CollecTor are Onionoo[6] and MetricsWeb.[7] Onionoo aggregates the descriptors available at CollecTor and provides current and historic data about *currently running* Tor servers. Based on Onionoo there is an ecosystem of clients building visualizations and other results helping users to find the piece of information they need.[8]

MetricsWeb uses CollecTor's data for providing *the history* of the entire Tor network in form of aggregated and enriched data sets, which serve as the basis for the numerous visualizations on MetricsWeb and can be freely downloaded for further use.

2.2 Privacy goals

The goals of a privacy and anonymity network like Tor are not easily combined with extensive data gathering, but at the same time data is needed for monitoring and improving the network and detecting possible censorship events or attacks against the network. Safety and privacy concerns regarding data collection by Tor Metrics is guided by the Safety Board's guidelines.[9] Safety and privacy assessment is usually done informally by discussion during the proposal process[10] for changes to the Tor source, and/or supported by closer analysis in form of Tor Tech Reports, for example, the introduction of onion service statistics was backed by a Tor Tech Report [8], which substantiated the privacy standards implemented and the statistical accuracy of the data to be collected.[11]

It is out of scope of this report and will be future work to provide such an assessment for both privacy and statistical accuracy throughout the data-verse of Tor Metrics. Until such

[3] See [1, dir-spec.txt] about how to retrieve descriptors.

[4] The main instance is https://collector.torproject.org. Since 2016 there are also several mirror instances sharing their data to gather even more of the ephemeral descriptors and other Tor network related data.

[5] Actually, some data, e.g., bridge descriptors, are pre-processed in order to remove possibly privacy critical information, but for the current report they can be still considered raw data.

[6] https://oninoo.torproject.org

[7] https://metrics.torproject.org

[8] See https://metrics.torproject.org/development.html for development tools and https://metrics.torproject.org/operation.html for user centered services.

[9] See https://research.torproject.org/safetyboard.html#guidelines.

[10] The proposal process is defined in [1, proposals/001-process.txt] and security and anonymity implication should be part of any proposal (cf. [1, line 114 of proposals/001-process.txt]).

[11] See also the related blog post https://blog.torproject.org/blog/some-statistics-about-onions.

background is available security and privacy assessment will be based on the guidelines, best practices, and heuristic arguments. The current report focuses on in-memory data and considers the scenario that an attacker gains access to in-memory storage. Thus, any run-time data for normal processing as well as the in- and outgoing traffic are also available to the intruder. Hence, at most events/data that occurred and were finalized *before* the breach can potentially be protected.[12] Another goal is to reduce reporting of potentially privacy problematic data.

2.3 Measuring and counting

Tor instances keep data in-memory and on disk for normal operation, for facilitation of local administration of the Tor server, and for reporting metrics data. The latter is mainly accomplished by uploading extra-info descriptors to authorities. For a quick orientation about the structure of these descriptors two examples of extra-info descriptors can be found in the appendix on page 24.

2.3.1 Server internal processing

Servers write their measurements and counting results to separate files, the "stats files", which are located in sub-folder `stats` of a configurable path. These files are parsed and their content is assembled to form an extra-info descriptor, which will be uploaded to an authority. The upload of extra-info descriptors happens together with the upload of the server descriptor.

The callback `check_descriptor` runs every minute, checks, if descriptors have to be uploaded, if necessary, it creates the server descriptor and the extra-info descriptor, which is populated from previously prepared stats files: dirreq-stats, hidserv-stats, entry-stats, buffer-stats, exit-stats, conn-stats.

Writing of stats files is triggered by two callbacks, `write_stats_file_callback`[13] and `record_bridge_stats_callback`.[14] These callbacks are registered to be run regularly after their first start after one second.[15] Afterwards, the corresponding tasks are run in their own intervals, i.e., after running for the first time the next interval is currently limited to maximal one hour and the actual interval will be the smallest demanded by the respective sub-tasks. Given that all configuration options for statistics are enabled the following functions are called from `write_stats_file_callback`:

`rep_hist_buffer_stats_write`: statistics about cell processing for monitoring relay performance (cf. 2.4.6)

`geoip_dirreq_stats_write`: directory statistics (cf. 2.4.1)

`geoip_entry_stats_write`: entry contact statistics (cf. 2.4.2)

`rep_hist_hs_stats_write`: onion services statistics (cf. 2.4.7)

[12] The goal that an adversary cannot learn the state of the measurement before time of compromise, is usually referred to as *forward privacy*.

[13] As defined in [13, `src/or/main.c:1702-1747`].

[14] As defined in [13, `src/or/main.c:1752-1777`].

[15] Scheduled by the code in [13, `src/or/main.c:1265-1275`] and the callback's return values.

`rep_hist_exit_stats_write`: exit traffic statistics (cf. 2.4.8)

`rep_hist_conn_stats_write`: traffic statistics between relays (cf. 2.4.9).

`rep_hist_desc_stats_write`: statistics about served descriptors (only for bridge authorities).

The `record_bridge_stats_callback` only triggers one function: `geoip_bridge_stats_write`, which writes bridge connection statistics (see 2.4.2).

All of these functions verify, if their individual measurement interval has elapsed. If so, they assemble their respective data, reset the data collecting structures, and write the data to files in the configured statistics directory. This process is similar for all stats-files, but not identical. Some of the concerned functions handle the reset of the measurement structures in-memory immediately after assembling the data to be written and others only reset after a successful write. For example, `geoip_entry_stats_write` only resets the data structure when writing succeeds, which can cause data retention for more than the intended 24 hour interval and `geoip_bridge_stats_write` doesn't remove client IPs from memory until the *next* interval's statistics are going to be written, which leads to a usual retention time of up to 48 hours.[16] In `geoip_entry_stats_write` the removal of older client data is only performed, if the interval for the next reporting is reached,[17] and `geoip_remove_old_clients` removes clients older than the current report interval of 24 hours, which is the argument `start_of_dirreq_stats_interval` and then removes the data after computing and writing statistics.[18] Thus, if writing fails there could be up to 48 hours of client data available in-memory.[19] For bridge clients ip connections the retention time is usually more than 24 hours, because the old clients are removed[20] *before* statistics computation and here only those from *before* the current reporting interval.

2.4 Data structures

This section describes in-memory storage structures for all data collected for metrics purposes and explains how these structures are maintained during a measurement interval.

The following assumes some familiarity with the data fields of extra-info descriptors.[21] The descriptions are grouped by the extra-info descriptor target field and exclude fields that are not in the focus of this analysis, e.g. identity, digests, statistic interval end times.

2.4.1 Directory requests counts

In order to derive usage by country Tor servers keep track of the originating country of directory requests. The resulting data is written to extra-info field `dirreq-v3-reqs` as a list of mappings from two-letter country codes[22] to the number of requests for v3 network statuses from that

[16] Some of these difference were introduced on purpose, e.g., the 48 hour interval seems to be due to a technical choice for bridge metrics, as it is already mentioned in the introduction of the extra-info proposal cf. [1, proposals/166-statistics-extra-info-docs.txt].

[17] See [13, `src/or/geoip.c`:1627-1654].

[18] [13, `src/or/geoip.c`:1648]

[19] Cf. [13, `src/or/geoip.c`:1644,1645]

[20] In function `geoip_bridge_stats_write` [13, `src/or/geoip.c`:1492-1530].

[21] Two example descriptors are printed in appendix A.2.

[22] GeoIp codes usually refer to countries, but in some cases to other kinds of jurisdiction. For the topic treated in this report it does no harm to simply refer to countries in all cases.

country, rounded up to the nearest multiple of 8.

During run-time the counts are stored in a list of `geoip_country_t` structures[23] without binning or obfuscation. The count `n_v3_ns_requests` is increased when a client is recorded.[24] The map of `geoip_country_t` structures is reset[25] after writing the derived values to the stats file.

2.4.2 Connecting client counts

Connecting clients use the Tor network and their count is tracked in regard to originating country and in case of bridges also the transport used and the IP version. The resulting data is written to the fields `bridge-ips`, `bridge-ip-transports` and `bridge-ip-versions` as well as `dirreq-v3-ips` and `entry-ips`, of which the latter two are currently not used in Tor Metrics.

In order to avoid repeated counting of the same client IP connecting the client IPs are stored in-memory in maps of `clientmap_entry_t`[26] without binning or obfuscation.

The data reported in `bridge-ips` is used for all MetricsWeb graphs about bridge user counts and together with `bridge-ip-transports`, which is a list of mappings from pluggable transport names to the number of unique IP addresses that have connected using that pluggable transport, for MetricsWeb's *Bridge users by transport* and *Bridge users by country and transports* graphs. The values from `bridge-ip-versions`, which is a list of unique IP addresses that have connected to the bridge per protocol family, are used for MetricsWeb's *Bridge users by IP version* graph.

All the values above are reported rounded to the next multiple of eight. The counts are taken from the clientmap, binned, and written to the file `stats/bridge-stats`. All countries with at least one count are reported.

2.4.3 Directory response counts

Another field used to derive client contacts is `dirreq-v3-resp`, from which the success count of responses made by the Tor server is currently used to determine the client count of bridges. Field `dirreq-v3-resp` reports a list of mappings from response statuses to the number of requests for v3 network statuses that were answered with that response status, rounded up to the nearest multiple of eight. All response statuses with at least one response are reported.

Counts by response status are stored in a simple array without obfuscation[27] and the binned values are computed just before writing statistics to file `stats/dirreq-stats`, the array is reset after writing the statistics file successfully.[28]

2.4.4 Server bandwidth metrics

The fields `write-history` and `read-history` declare how much bandwidth the Tor server has used recently. Usage is divided into intervals of currently four hours. The end of the most recent

[23] As defined in [13, `src/or/geoip.c:55-59`].
[24] This happens by calling function `geoip_note_client_seen` in [13, `src/or/geoip.c:560-613`].
[25] See function `geoip_dirreq_stats_write` in [13, `src/or/geoip.c:1284-1312`].
[26] Defined in [13, `src/or/geoip.c:475-491`].
[27] Array definition [13, `src/or/geoip.c:640`] and array processing [13, `src/or/geoip.c:644-656`].
[28] In function `geoip_reset_dirreq_stats` cf. [13, `src/or/geoip.c:1179-1208`].

interval of the measurements is given. Values are the number of bytes used in the last intervals, ordered from oldest to newest. Stored in struct `bw_array_t` using circular arrays for maxima and totals.[29]

Similarly the extra-info descriptor fields `dirreq-write-history` and `dirreq-read-history`[30] declare how much bandwidth the Tor server has spent on answering directory requests. These values are cut at the value of the configured max bandwidth for reporting. They are also stored in struct `bw_array_t` (as `write-history` and `read-history`).

All four `*-history` values are stored without obfuscation or binning and are only cutoff and rounded down to 1KB before they are reported.[31]

2.4.5 Directory download metrics

`dirreq-v3-direct-dl` and `dirreq-v3-tunneled-dl` provide statistics about possible failures in the download process of v3 network statuses. The list currently contains values for `complete`, `timeout`, and `running`. Values are stored in a map of `dirreq_map_entry_t` types.[32]

The values are rounded to the next multiple of 4 before printing statistics and only printed when the rounded value of `complete` is bigger than 16. After writing stats the values are cleared.[33]

2.4.6 Circuit metrics

`cell-*` Data is derived from circuits[34] held in-memory for normal operation. The values are derived at report time and statistics for disposed circuits are stored at the time of their disposal. After assembling the data, which will be written to the `buffer-stats` file, the data structure used is reset.

2.4.7 Onion services metrics

Onion services metrics are reported mainly in two fields: `hidserv-rend-relayed-cells` and `hidserv-dir-onions-seen`. `hidserv-rend-relayed-cells` reports the approximate number of relay cells seen in either direction on a circuit after receiving and successfully processing a rendezvous cell. The original measurement value is obfuscated only for reporting[35] and stored in-memory as part of the `hs_stat_t` structure without binning or obfuscation.[36] `hidserv-dir-onions-seen` reports the approximate number of unique onion-service identities seen in descriptors published to and accepted by this onion-service directory. The original measurement value is obfuscated only for reporting,[37] whereas this value is derived from the

[29] Cf. [13, `src/or/rephist.c:1209-1236`]
[30] Assembled in [13, `src/or/rephist.c:1497-1550`]
[31] The in-memory values are not changed, cf. function `rep_hist_fill_bandwidth_history` in [13, `src/or/rephist.c:1448-1491`].
[32] [13, `src/or/geoip.c:700-714`]
[33] Cf. footnote 28, page 6.
[34] See `circuit_t` in [13, `src/or/or.h:2943-3084`].
[35] Cf. [1, section 2.1.2 of dir-spec.txt]
[36] Cf. [13, `src/or/rephist.c:3002-3009`]
[37] See footnote 35, page 7.

hs_stat_t structure, which contains a clear list of digests of the onion services' public keys. The in-memory struct is reset after creating the report string for the statistics file hidserv-stats.[38]

2.4.8 Exit traffic metrics

The fields exit-streams-opened, exit-kibibytes-written, exit-kibibytes-read contain information about exit traffic. Data for all three fields are kept in arrays.[39] The exact values for all ports are stored in-memory. The reported number of opened exit streams to a port is rounded up to the nearest multiple of four, the other two values are rounded to the next 1024 bytes.[40] All in-memory counters are erased after computing the metrics.[41]

2.4.9 Connection metrics

The conn-bi-direct line is filled from simple counters.[42] The data reported is used for network and relay related statistics, which are provided by MetricsWeb as one of the performance related graphs. The counters are reset immediately after statistics computation independent of write success.[43]

2.4.10 Unused extra-info descriptor fields

The data of the following extra-info descriptor fields are currently not used anywhere in Tor Metrics:

- all cell-* fields,
- all exit-* fields,
- dirreq-v3-direct-dl and dirreq-v3-tunneled-dl.

It might be a premature decision to simply stop reporting these unused values in extra-info descriptors, because the reason for not using them could be lack of awareness that they are reported or a lack of resources to put them to use. For example, the values from exit-node related fields, i.e., exit-*, could be used to address questions related to exit data, which are asked in research (e.g. in [7] cf. 4.2), and to introduce new statistics and graphs in MetricsWeb as well as making aggregate data sets available. On the other hand, concerns where raised that providing the exit-* statistics would enable attacks that could uniquely identify the applications used by clients or for fingerprinting unusal port etc.[44] In general, ending the collection of currently

[38] Reported metrics of onion services are binned and obfuscated using the Laplace distribution. The exact parameters are defined in [13, src/or/rephist.c:3112-3133].

[39] Cf. [13, src/or/rephist.c:2067-2072]

[40] The calculation of the reported values is performed in rep_hist_format_exit_stats [13, src/or/rephist.c:2120-2269]

[41] Cf. [13, src/or/rephist.c:2291].

[42] Cf. function rep_hist_format_conn_stats [13, src/or/rephist.c:2903-2922].

[43] Cf. function rep_hist_conn_stats_write line 2942 [13, src/or/rephist.c:2928-2952].

[44] These were brought to Tor Metrics attention by Rob Jansen who addressed the topic in Tor Metrics' mailing list: "Tor is classifying its traffic into ports, which could uniquely identify the application being used by the client. They also track bandwidth usage per port (and per exit); again, this is bad for those using a random or unique looking

unused data should be considered carefully and not hastened. The future assessment of all Metrics' data will be the right project to address the question of whether to keep or drop the collection of currently unused metrics.

2.4.11 Other data

This section concentrates on data gathered or written for other purposes than filling an extra-info descriptor.

Stats Heartbeat

The function `log_heartbeat`[45] performs some checks to determine the state of the running relay/bridge, but also logs some statistics about client connections. `log_heartbeat` is one of the periodic event callbacks.[46] Function `format_client_stats_heartbeat`[47] computes the exact number of different client connections for the last six hours using `client_history` unless turned off or set to a different interval in property `HeartbeatPeriod`. In addition, the number ob bytes written and read by the Tor server process is logged.[48]

Logging

The debug level logs client data in addition to operational data.

`geoip_note_client_seen` logs the client seen with the transport used in debug mode.[49]

`geoip_get_transport_history` logs the true total number of clients and the true numbers for each transport type in debug level.[50]

`rep_hist_note_exit_bytes` logs for each port the true number of bytes read and written in debug mode.[51]

`rep_hist_note_exit_stream_opened` logs the port to which an exit stream was opened in debug mode.[52]

---

ports (that a given exit does not see very often) because it could be used to create a fingerprint. Intersection attacks become easier with this information."

`cell-*` statistics are percieved less critical, but still: "This provides queue timings and number of cells being processed at a relay. The number of cells can be used to compute bandwidth of circuits. It may be possible to launch some attacks that create several circuits with the intent of moving which decile buckets some legitimate circuits get placed into, but this is less worrisome of an attack than the others."

[45] [13, src/or/status.c:91-165]

[46] [13, `src/or/main.c:1193-1220`]

[47] [13, `src/or/geoip.c:1457-1488`]

[48] Cf. [13, `src/or/main.c:159-162`].

[49] See line 582 in `geoip_note_client_seen` (cf. footnote 24, page 6).

[50] Cf. [13, `src/or/geoip.c:863,885,900`].

[51] Cf. [13, `src/or/rephist.c:2313`].

[52] Cf. [13, `src/or/rephist.c:2325`].

3 Possible privacy issues

Tor servers configured to keep statistics and report extra-info descriptors[53] have a reporting interval of 24 hours. The following types of data are held in-memory up to this interval or even longer depending on the type of data and time of collection.

- Client IPs from various types of contacts to a server, i.e., contacts to bridges, to entry relays, to directory mirrors.
- Public key digests of onion services and cell counts (cf. 2.4.7);
- bandwidth used generally and bandwidth consumed for serving directories (cf. 2.4.4);
- exit traffic stream count as well as exit bytes written and read (cf. 2.4.8).

The most critical data in the above list are client IPs and related information.[54]

The following extra-info fields depend on code and in-memory structures used for storing the client IPs:

- Provided by bridges:
  - unique client count by country of origin for every contact in field `bridge-ips`,
  - IP version in `bridge-ip-versions`, and
  - transport used in `bridge-ip-transports`.
- Relays and bridges report:
  - unique client count by country of origin for directory requests `dirreq-v3-ips` for successful responses.
- Entry guards report `entry-ips`, i.e., the unique client count by country of origin for every contact.

Some of these fields, namely `dirreq-v3-ips` and `entry-ips`, are currently not used further up in the Tor Metrics data processing chain, but others support vital client statistics about the Tor network.[55] Section 4 explores the options for keeping these statistics and reducing or even avoiding the in-memory storage of lists of IP addresses of Tor clients.

[53] Reporting of extra-info descriptors can be turned off or limited via configuration. It is assumed that reporting and logging options are enabled, i.e., Tor server options like `BridgeRecordUsageByCountry`, the various `*Statistics` etc. are set to 1.

[54] These were also mentioned as most critical by Rob Jansen in his mail to Tor Metrics' mailing list:

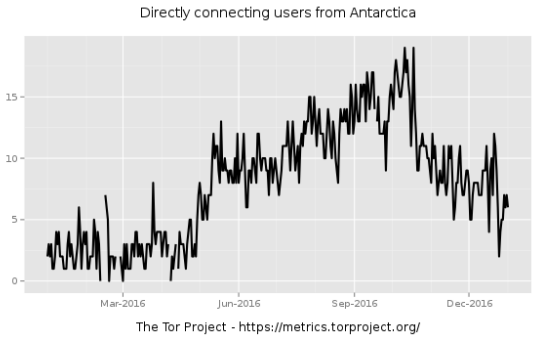
"[unique ips per country code] *-ips (there are many of these, e.g. "entry-ips") Usually this involves storing individual user IP addresses in memory (in order to track uniqueness) over some period of time (usually 24 hours), sometimes for longer than the user would have otherwise been known to Tor (if a user's session is 1 hour, Tor could remember the IP for at most 23 additional hours). This is reported, e.g., per entry; there are many cases in the data where it is very likely that only one user is connecting to a guard from a given country (because it is rounded up to 8). Users in small countries have the greatest risk (intersection attacks become really easy)."

[55] The bridge client count estimates are built on `bridge-ips`, `bridge-ip-versions`, `bridge-ip-transports`.

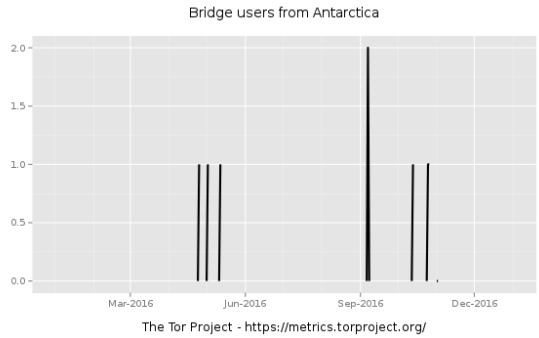
The client data sets of MetricsWeb and the visualizations based on them occasionally cause questions about privacy implications of small client counts per country or per country and transport. Section 3.1 gives some examples and provides information about client counts per country and other parameters. These concerns are raised for tables and graphs at the aggregated data level, but the underlying data is tightly connected to the IP addresses collected in-memory. Thus, it makes sense to also address this privacy issue in the current report, which is done in section 4.3.4.

3.1 Small clients counts

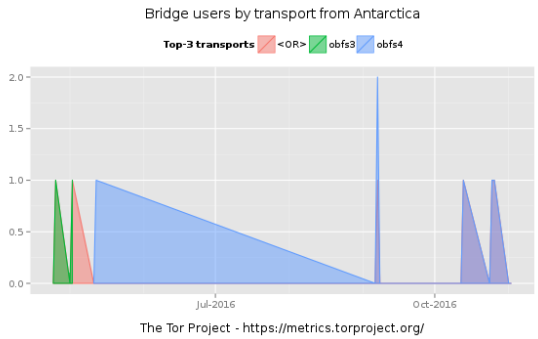
Small countries usually have very tiny Tor client counts, examples for Antarctica and Vatican City are shown figures 1 (page 11) and 2 (page 12).



(a) Users connecting to relays.



(b) User(s) connecting to bridge(s).

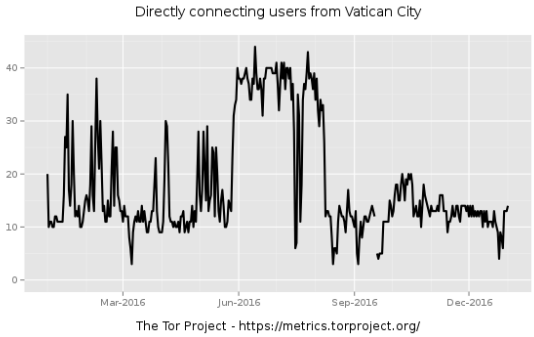


(c) User by transport.

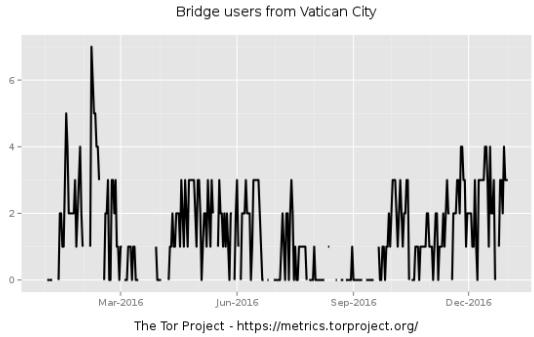
Figure 1: Antarctica Tor usage 2016. MetricsWeb is the source of all graphs (see table 2).

Counts of clients from Antarctica directly connecting to the Tor network during the year 2016 are graphed in figure 1a, the even smaller count of bridge users from Antarctica in 2016 in figure 1b, and figure 1c breaks the bridge connection down into the type of transport used. Similarly Tor client count during 2016 for Vatican City. Most notably, bridge users from Antarctica (1c) and Vatican City (see figure 2c) seem to be all distinguished by the type of transport they use.

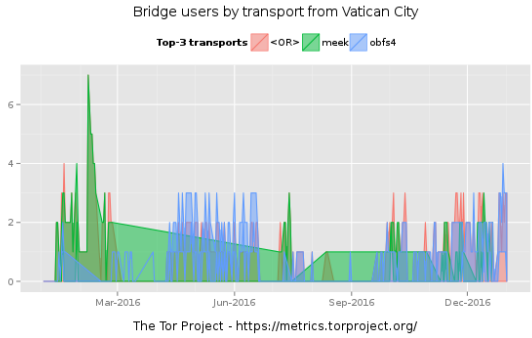
These two are not even the most extreme examples in terms of client counts, Vatican City has a median of 13 users in 2016 and Antarctica a median of 8. For 2016 there are 25 countries with a median user number less than ten. Table 1a (page 23) shows the count of countries with



(a) Users connecting to relays.



(b) User(s) connecting to bridge(s).



(c) User(s) by transport.

Figure 2: Vatican City Tor usage 2016. MetricsWeb is the source of all graphs (see table 2)

less than *m* median users per day in 2016, and as contrast table 1b lists the count of countries with median user numbers starting at 1000.

These small counts of distinguishable subsets of Tor clients look problematic concerning privacy.[56]

4 Mitigate privacy impact

The following sections take a look at various techniques/mechanisms/systems to reduce privacy impact reaching from privacy aware counting in 4.1 over using Tor external data gathering systems in 4.2 to exploring the options of simply avoiding the collection of problematic data in 4.3.

Many of the techniques and measurements listed in the following sections are far from being implemented and would need extensive work to be useful in practice. Hence, the following should be read as a description of what might be possible and not as what will be implemented in the near future. A more concrete list of what could be implemented in the nearer future is given in sections 4.4 and 6.

[56] Also cf. footnote 54, page 10.

4.1 Counting, surveys, sketches

Counting of unique items is naïvely done by keeping a unique list of these items. For finding an approximate count of unique items this could be avoided trading in accuracy of the resulting metrics. The following sections discuss mechanisms for counting without keeping all items in-memory.

4.1.1 Probabilistic counting

Estimating the count of unique items, e.g., connecting clients, without storing all items registered during the measurement interval could be solved by probabilistic counting as proposed in [4]. Without any additional randomization this would give a part of the clients additional privacy by plausible deniability depending on the used hash function[57] and certainly provide another barrier for an attacker to determine client IPs from the data held in-memory. Compared to the current scenario this could provide a gain in privacy for the IP counting task. In addition, error estimates and efficiency of the probabilistic counting method are known (see [4]) and would provide a basis for computing the aggregate statistics from the individual reports.

The steps necessary for deploying such a solution require extensive effort: for the actual implementation the hash function used and size of sketches as well as the accuracy of the count estimate need to be chosen. The intended accuracy leads to the decision between using the simple algorithm or the algorithm with stochastic averaging. The metrics derived might need to be adjusted depending on the now available error estimates.

4.1.2 Privacy preserving surveys

Clients connecting to a Tor server could be viewed as entities taking a survey. A recent approach with even differential privacy guarantees[58] is the method proposed in [3, RAPPOR - Randomized Aggregateable Privacy Preserving Ordinal Response]. RAPPOR is based on client side generation of noisy sketches and a machine learning approach for evaluating these sketches to calculate estimates for the statistics of interest. Clients need to compute an initial noisy sketch from their data, which is called permanent randomized response, and use this permanent response to produce an again obfuscated sketch, the instantaneous response, as actual report. The instantaneous response sketch would have to be part of all those connections made by the client that are used for statistics, e.g., it would need to be added to a directory request.

In total, the changes necessary for implementing a protocol like RAPPOR are extensive: changes to the client code, the Tor server code, the communication protocol, and the final processing for deriving the wanted estimates. A survey setting trusting client generated data sketches would also open room for spam or manipulation of the metrics taken.

[57] But it cannot prevent the identification of certain IPs with high probability (for example, cf. [6, section 4.1.1] or [15, section 2.2]).

[58] See [3, section 3] for definition and proves of their differential privacy claims for RAPPOR.

4.2 Metrics systems proposed by Tor related research

With the progress of privacy research during the last years metrics systems for collecting network data in a privacy conscious manner were proposed. Two systems explicitly targeting metrics collection from the Tor network, which not only provide the design and privacy assessment of their system, but also make the code-base from their respective proof of concept and reference implementations freely available, are PrivEx [2] and PrivCount [7]. It is out of the scope of this report to suggest or discuss any replacements or additional metrics systems for the current file based Tor Metrics system. Still, looking closely at PrivEx and PrivCount provides valuable insight about what they deem potentially privacy endangering data and what data of interest might not yet be available through Tor Metrics.

PrivEx [2] proposes a metrics system running separately from Tor instances and introduces its own network of various types of server instances. The data processed is retrieved from adapted Tor server instances via the controller protocol, which is extended for PrivEx purposes.

PrivCount builds on one collection scheme introduced by PrivEx and extends its collection ability as well as some operational properties. The data collecting instances of the PrivCount network also use the controller protocol, i.e., an extended version of the currently implemented protocol, to retrieve the data of interest from the Tor server they are collecting from.

The main purpose of PrivEx' reference implementation is the combination of in- and outgoing traffic of the Tor network. In particular, identifying the number of connections made from Tor clients to possibly censored web-addresses, which gives an estimate about Tor usage for censorship circumvention.

PrivCount focusses on entry and exit statistics. This comprises client counts at the entry nodes, which are collected via the extended controller protocol and not based on the Tor server internal client IP list, and various metrics for traffic exiting the network. PrivCount's exit statistics are concerned with streams exiting via certain ports and the influence of exit policies on exit traffic.[59]

In general, an externally operated metrics system is quite expensive to maintain compared to the current Tor Metrics system. Furthermore, newly implemented controller events for retrieving data could be also a data source for an attacker, if not properly secured by the server operator. It would require additional operation of metrics server instances, additional maintenance of the code-base, and additional processes to integrate the new data sources into the existing ones. In addition, the privacy properties of such system and the security of their implementation would be more difficult to assess from external parties than the current descriptor based Tor Metrics system.

4.3 Mitigating implementation changes

The following measures are directly derived from source code analyses of both the metrics related Tor server code and the core Tor Metrics code for data aggregation and client count estimation. They are generally concerned with avoiding data gathering and reducing the availability of sensitive data via other channels like logging or controller events.

[59] The authors of [7] don't address why the data provided in the various extra-info descriptor fields `exit-*` is insufficient or how the data overlaps.

4.3.1 Reduce duration of in-memory data retention

Tor servers configured to report statistics keep client IP addresses and associated information in-memory for at least one measurement interval of 24 hours. Unfortunately, the current code retains these IPs and related information for even up to two such measurement intervals (in case of bridges), because the old data originating from the previous interval is only released before writing statistics about the current measurement interval. Erasing data immediately after computing statistics would more than half the retention time.

4.3.2 Avoid problematic logging and controller events

Some of the possibly harmful data held in-memory for providing metrics is currently also used for logging and responding to controller clients.

The controller protocol is defined in [1, control-spec.txt] and allows triggering of the heartbeat log message ([1, section 3.7, control-spec.txt]). Another request defined in [1, sections 3.9 and 4.1.14, control-spec.txt] to receive information from bridges about recent client connections.[60] The replies contain complete counts by country and transport (also see `geoip_bridge_stats_write`).

Using the option `HeartbeatPeriod` a Tor server can be configured to write a recurring log message, which serves the purpose of informing the operator that the server is still running and working. The minimal reporting interval is 30 minutes and the statement logged contains the exact number of different client connections for the last six hours. In addition, the heartbeat log message can be triggered (without any time constraints) by a controller client signal.

Additional logging of collected data, e.g., client counts per transport, exit port opened and exit bytes read/written, takes place in debug mode (cf. 2.4.11).

In order to improve Tor client privacy these functionalities ought to be changed to only report data unrelated to client IPs and only about time intervals equal or bigger than the chosen reporting intervals for extra-info descriptors.

4.3.3 Replace problematic data sources

Client IPs are currently only kept in-memory for deriving estimates of bridge client counts where at the same time the estimates for direct Tor client counts are derived from counts of successful directory requests taking multiple requests into account as these occur usually as a constant factor for each client. There is no reason, why the estimations should differ and the IP lists in-memory became obsolete, if the same estimation method for bridge client counts would be supplied.[61] This would cause the estimates to be even more comparable and also reduce configuration and simplify the metrics related code of the Tor.

Such a removal would affect the following extra-info descriptor fields:

- `dirreq-v3-ips`,
- `entry-ips`,

[60] In particular this is the clients-seen event, which is used by nyx `arm.torproject.org`.

[61] The question about removing the map and corresponding measurements from the code that hold client IP addresses was raised a while ago, for details see Tor Bugtracker [14, ticket #15469].

- `bridge-ips`,
- `bridge-ip-versions`, and
- `bridge-ip-transports`.

As all extra-info descriptor fields regarding entries and bridges are concerned, the two fields `bridge-stats-end` and `entry-stats-end` would loose their meaning and could also be omitted.

The fields `dirreq-v3-ips` and `entry-ips` are currently not used for any statistics or data sets provided by Tor Metrics and could be dropped.

All other fields from above are the basis for bridge client count estimates.[62] The field `dirreq-v3-reqs` is also available in extra-info descriptors uploaded by bridges[63] and could be used for clients by country count for replacing `bridge-ips`. The fields `bridge-ip-versions` and `bridge-ip-transports` are used to estimate fractions of the client counts that have their origin in a certain country or use a certain IP version. These could be filled by counting countries and versions of the occurring requests registered in `dirreq-v3-reqs`, of course the corresponding aggregated statistics and estimates need to be adapted. All fields mentioned above could be dropped and two new fields for both relay and bridge extra-info descriptors need to be added; in particular, `dirreq-v3-versions` and `dirreq-v3-transports`.[64] This would lead to less fields in extra-info descriptors, increased privacy, and provide more comparable estimates for relays and bridges. A more detailed description and analysis of the included processing changes for generating estimates is given in section 5.

4.3.4 Obfuscate stored and reported data

Client counts per country can be very low on a server basis, e.g., roughly 80% of counts reported in extra-info descriptors for the three biggest Tor user groups (de, ru, us) only report the lowest count possible. Raising the available threshold constants for reporting total client counts and client counts by country[65] cannot be used as mitigation measure as it also would render most of the client count estimates useless. Instead of using thresholds a white list could be introduced that lists all countries for which the count should be recorded. Only countries on the white list would be added to the counting array and all others would be obfuscated by summing them under `other`. The list itself could be provided in an easily parsable text format added to Tor server source code.

There are two ways to choose countries for the white list: either by population size or by Tor usage based on Tor client count statistics. A choice by population count at a threshold of $2,000,000$ would lead to a list of 147 white listed countries.[66] Using the Tor usage approach a cut-off at a daily mean of 1000 Tor clients would generate a list of 97 countries based on data from 2016 (cf. table 1b).

[62] For details see section 5.1. The directly connecting client count is entirely based on `dirreq-v3-reqs`, which is not derived from a clientmap structure.

[63] In 2016 roughly 80% of bridge extra-info descriptors that provided `bridge-ips` also contained `dirreq-v3-reqs`.

[64] For consistent naming it might be useful to change the field name `dirreq-v3-reqs` to `dirreq-v3-countries`.

[65] As defined in [13, `src/or/geoip.c:658-667`].

[66] According to the World Factbook [16, countries by population size and raw data].

Either choice of generating the white-list would need to be re-adjusted yearly or more often, which would also cause additional maintenance work. The second approach would be more difficult to adjust, because once a white-listing mechanism is introduced the data for adjusting won't be available anymore from Tor Metrics statistics and would need to be generated by other means.

It should also be evaluated, if client directory responses (field `dirreq-v3-resp`) and the client count related fields proposed in section 4.3 (client counts by country `dirreq-v3-countries`, version `dirreq-v3-versions` and transport `dirreq-v3-transports`) even when not based on in-memory client IP lists should be obfuscated. In order to obtain obfuscation for both the in-memory counts and the reported results noise addition at counter initialization seems to be an efficient measure on first glance. For onion service statistics Tor Metrics implemented the generation of Laplace noise,[67] which could be applied in the current scenario and fosters code reuse of critical parts like the Laplace noise generation.

But, a simulation applying noise to collected data and processing the resulting data further for use in MetricsWeb showed that the additional noise would render the existing statistics very inaccurate.[68] Thus, it is advisable to conduct further research and wait for the already planned assessment for both privacy and statistical accuracy throughout the data-verse of Tor Metrics.

4.4 Conclusion

Integrating counting systems or parts thereof as discussed in section 4.1 would require extensive design and implementation work for changes of the current Tor source code and also for the aggregating and estimation code further up in the Tor Metrics processing chain.

Applying the measurement systems outlined in section 4.2 in Tor Metrics would mean a step toward using a second totally different and more costly manner of measuring Tor. If introduced in addition to the current file based system (cf. 2) the cost of operation would be very high and the actual problem of in-memory retention is not addressed as the Priv* systems use Tor's internal accounting of client connections and other measured data.

The changes necessary for the third approach in 4.3.3 affect both the Tor server and Tor Metrics code bases in very clear ways, which consist mostly in code reduction, streamlining, and using different fields of already parsed extra-info descriptors. Thus, it seems to be the most feasible answer for improving privacy current Tor server code. The other measures listed in section 4.3 would easily fit into the changes necessary for applying 4.3.3 or be obsolete with the introduction of these changes.

The details and various steps of avoiding and reducing data collection are given in section 5.

[67] [1, proposals/238-hs-relay-stats.txt] and [5, 8]

[68] A closer look at the involved statistics: The current estimations for user counts by country rely on a sum of reported data. In the sketched obfuscation scenario this sum would also contain a sum W_n of Laplace random values, where n is the number of reported values for the particular country. The standard deviation of W_n depends on the obfuscation parameters and on $\sqrt{n}$. A daily median (mean) of reports from relays is around 1300 (mean: 2200) and 550 (mean 150) for bridges. Such values are not tolerable in the current estimation process and obfuscation should only be introduced with additional measures to keep the existing accuracy.

5 Impact of implementation changes

Section 4.3.3 sketches a solution for avoiding the in-memory storage of client IP for client count metrics by replacing the source of vital estimates. First the actual methods for client count estimation are discussed in 5.1. Based on this the changes necessary are detailed in 5.3 before identifying the changes to the Tor server code and the possible side effects in 5.4.

5.1 Client related estimates

The current method of estimating client numbers was introduced in 2013 for both bridge and relay clients[69] to replace an estimation method based on the number of unique IP addresses making connections to Tor servers. The daily estimate uses values taken from extra-info descriptors, in particular the count of daily directory responses (respectively requests) and the number of bytes written delivering the directory data.[70] According to [10] it suffices to estimate the total number of directory requests to bridges and relays, from which the client count is calculated directly.

The data from extra-info descriptors used for bridge related estimates is also available for relays. Thus, it seems natural to apply the same formula for estimating relay client numbers. Looking at the code the implementations differ for bridge and relay clients. For relay clients the code diverts from the estimation method explained in [10] and uses request counts per country.[71] The raw data is taken from extra-info descriptor field `dirreq-v3-reqs` and used to fill clients by country counts as well as the entire count of clients for this relay.

Bridge client counts are implemented as suggested in [10] and are estimated from directory request responses as well as contact IP counts,[72] which are derived from `dirreq-v3-resp`. The total value of client contacts is taken from descriptor field `dirreq-v3-resp` (the successful responses) and counts of connections from different countries is derived from the field `bridge-ips`. The fractions for `version` use descriptor field `bridge-ip-versions` and `transport` is derived from `bridge-ip-transports`. The bridge client number estimates per country build on the estimate for the number of total clients and derive the client numbers by applying the fraction per country estimated from the number of connections made by country, i.e., `bridge-ips`. The current method of estimating caused unlikely results for the number of bridge clients by country. The discussion and analysis of these problematic results[73] suggests that switching to a calculation of bridge client count estimates that uses the same extra-info descriptor fields as direct client count estimates would even improve the estimate.

[69] See ticket [14, #8462] and related code https://gitweb.torproject.org/metrics-web.git/log/?qt=grep&q=8462. The code was integrated into Tor Metrics code during 2015.

[70] Values taken from extra-info descriptor fields `dirreq-v3-reqs`, `dirreq-v3-resp` and `dirreq-write-history`.

[71] Listed in descriptor field `dirreq-v3-reqs`. The relevant code can be found in [11, from line 91 of `modules/clients/src/org/torproject/metrics/clients/Main.java`].

[72] [11, starting at line 230 of `modules/clients/src/org/torproject/metrics/clients/Main.java`]

[73] A discussion via Tor Bugtracker [14, ticket #18167] began a year ago considering the usage of various fields for bridge client per country estimation, i.e., `bridge-ips` vs. `dirreq-v3-reqs`.

Most bridges report `dirreq-v3-reqs` already, for 2016 almost 90% of bridges reporting `bridge-ips` also provided the field `dirreq-v3-reqs` in their extra-info descriptor.

5.2 Data changes

The change proposed in 4.3.3 would result in dropping the fields `bridge-stats-end`, `bridge-ips`, `bridge-ip-versions`, `bridge-ip-transports`, `entry-ips`, `entry-stats-end`, and `dirreq-v3-ips` from extra-info descriptors. Two additional fields `dirreq-v3-transports` and `dirreq-v3-versions` need to be added in order to keep the current Tor Metrics statistics about client counts.[74]

5.3 Metrics changes

As explained in 5.1 the client count estimates for relays are already independent of descriptor fields that are to be dropped. Using the same estimation approach for bridges would lead to more comparable and even more accurate results (cf. 5.1). The necessary code changes for MetricsWeb would result in unified processing of the two extra-info descriptor types. The changes necessary for metrics-lib/DescripTor would in general result in providing the two new methods for the additional fields, but are free of changes to the parsing logic.

Another affected code base of Tor Metrics would be Onionoo, which uses the fields `bridge-ips`, `bridge-ip-versions`, and `bridge-ip-transports` for providing additional information in bridge *client documents*.[75] The relevant Onionoo protocol fields depending on `bridge-*` descriptor fields are still in *beta* stage and could either be removed or simply be filled from the new fields available, which is a minor code change.[76]

5.4 Tor server changes and side effects

When describing the code changes one needs to make choices; and the choice here was to describe the maximal code reduction possible, but of course there is room to alter the proposed changes and still reach the intended goal.

The following gives a terse overview of the code changes necessary for Tor server according to section 5.2. Also provided are possible Tor server configuration simplifications, and side effects or changes regarding logging and controller functionality.

5.4.1 Server changes

With the changes to extra-info descriptor proposed in 5.2 the Tor server options

- `BridgeRecordUsageByCountry` and
- `EntryStatistics`

could be omitted and replaced by option `DirReqStatistics`, which could be used for all types of servers alike.

[74] The naming is chosen along the current naming scheme that includes the string `v3`. It might be useful to drop this string from all of the `dirreq-*` descriptor fields.

[75] See section *Bridge clients objects* of the Onionoo protocol definition https://onionoo.torproject.org/#clients and Onionoo source code [12, class `org.torproject.onionoo.updater.ClientStatusUpdater`].

[76] Taking into account that the new fields would have multiple counts per day and client and would need to be adjusted with the factor used for the total client count estimation.

The alterations for the metrics providing code in Tor servers would mostly be code removal. The description follows the process of collection as described in 2.3.1 in order to cover all affected places in the server code.

The `record_bridge_stats_callback`[77] could be omitted entirely together with the following functions:

- `geoip_bridge_stats_init`,
- `geoip_bridge_stats_write`,
- `geoip_get_transport_history`,
- `geoip_get_client_history`, and
- `geoip_format_bridge_stats` (also see 5.4.2 for controller related changes).

The second main metrics callback `write_stats_file_callback` would be kept, but shortened to not provide entry statistics anymore. The affected functions would be:

- `geoip_entry_stats_write`,
- `geoip_format_entry_stats`,
- `geoip_reset_entry_stats`.

Other functions for handling `clientmaps`:

- `geoip_remove_old_clients`,
- `remove_old_client_helper_`,
- `geoip_get_client_history`.

In order to record versions and transports for bridges lists of new structs `geoip_version_t` and `geoip_transport_t` similar to `geoip_country_t` would need to be defined.[78] The function `geoip_note_client_seen` would need to be adapted to fill the new structures for recording client data. In addition, the code for handling client ip statistics would need to be removed and the code for filling the new lists of `geoip_version_t` and `geoip_transport_t` would need to be added.

Changes would also be necessary for `geoip_dirreq_stats_write`, which is called from `write_stats_file_callback`. This function would need to be adapted to omit writing the dropped descriptor fields and add the new descriptor fields derived from the above mentioned structures. Any calls to `geoip_note_client_seen` with action `GEOIP_CLIENT_CONNECT` could also be removed.

[77] [13, `src/or/main.c:1752-1777`]
[78] [13, `src/or/geoip.c:55-59`]

5.4.2 Affected controller events

Once `clientmap` structures and related code are removed from Tor server code the controller code also needs to be changed. The functions `format_bridge_stats_controller` and `control_event_clients_seen` would either need to be removed or adapted to the new structures for recording the counts.

Another affected controller function is `format_client_stats_heartbeat`, which would need to be adapted to not report the client counts by country anymore.

6 Summary

The previous sections of this report describe the Tor Metrics processing chain and the data provided by Tor Metrics with the aim to identify several ways to improve privacy regarding data held in-memory for clients of the Tor network. Possible mitigation measures are surveyed and the most feasible approach was detailed in section 5.

Many of the discussed improvements generate a workload for several future projects and some also need further research. Nevertheless, a recommendation for a list of first changes can be derived:

- Replace the current server internal counting mechanism in order to avoid holding client IPs in-memory. This leads to the immediate privacy improvement of not keeping client IPs in-memory for statistical purposes.
- Use `dirreq-v3-reqs` for client count estimation (for both bridges and relays, as suggested in 4.3.3). This would keep the statistics on client count as accurate as before without relying on client IP lists.
- Base the new fields `dirreq-v3-versions` and `dirreq-v3-transports` on the current counting mechanism used for `dirreq-v3-reqs`. This would also keep the statistics based on client count as accurate as before without relying on client IP lists.
- Remove controller protocol parts that rely on the old client count mechanism. This would avoid reporting privacy impacting data to the control port.
- Remove unnecessary logging of vital data or tie the logging to test-mode for avoiding privacy impacting data in logs.

These changes have a clearly defined scope and would result in privacy improvement. Identifying immediate changes for implementation and defining future changes for metrics collection is based on the following steps:

- Distill a change proposal for the Tor server changes chosen to be implemented.
- Provide several Tor server patches for the changes identified above.
- Provide patches for the necessary adaptions in the Tor Metrics processing chain.

- Assess privacy questions as raised in this report and statistical accuracy throughout the data-verse of Tor Metrics. Also assess the introduction of more obfuscation measures for various client counts without impacting estimation accuracy. In addition, the removal of unused data fields from extra-info descriptors (as identified in section 2.4.10) should be addressed and evaluated.

The assessment listed in the last item above is in part a consequence of this report, which is planned to start in the second half of 2017.

A Appendix

A.1 Tables

m	10	50	130	210	340
C	25	52	87	103	116

(a) Count $C = |\{\text{median(c)} < \text{m}\}|$ of countries with median of daily users in 2016 less than the given limit m.

m in 10^3	1	5	10	50	100	200	300
C	97	51	26	5	4	2	1

(b) Count $C = |\{\text{median(c)} > \text{m}\}|$ of countries with median of daily users in 2016 higher than the given limit m thousands.

Table 1: Count of countries with median of daily users in 2016. There are roughly 250 countries, and 433.5 is the median of the median daily client count of all countries in 2016.

Figure	Source
1a	https://metrics.torproject.org/userstats-relay-country.html?start=2016-01-01&end=2016-12-31&country=aq
1b	https://metrics.torproject.org/userstats-bridge-country.html?start=2016-01-01&end=2016-12-31&country=aq
1c	https://metrics.torproject.org/userstats-bridge-combined.html?start=2016-01-01&end=2016-12-31&country=aq
2a	https://metrics.torproject.org/userstats-relay-country.html?start=2016-01-01&end=2016-12-31&country=va
2b	https://metrics.torproject.org/userstats-bridge-country.html?start=2016-01-01&end=2016-12-31&country=va
2c	https://metrics.torproject.org/userstats-bridge-combined.html?start=2016-01-01&end=2016-12-31&country=va

Table 2: Graph source URLs.

A.2 Extra-info descriptor examples

A.2.1 Bridge extra-info descriptor

```
@type bridge-extra-info 1.3
extra-info Unnamed EF93668E48BD4F8DB9DF6D4CFCBF1A7BB5EC7CC2
master-key-ed25519 a3febLYkK9UmKf4PDhrw/cTefN1l5X0LsAt7BqdcrLM
published 2017-03-01 17:14:17
write-history 2017-03-01 14:10:28 (14400 s) 2253824,1248256,1308672,489472,592896,300560384
read-history 2017-03-01 14:10:28 (14400 s) 6366208,5633024,7112704,4847616,6200320,306330624
dirreq-write-history 2017-03-01 14:10:28 (14400 s) 1581056,683008,673792,36864,33792,662528
dirreq-read-history 2017-03-01 14:10:28 (14400 s) 56320,9216,4096,4096,2048,2048
geoip-db-digest C14DF5AE94101562DEACDD296278B0EFA3EA26E5
geoip6-db-digest A88A828020A558D37F97CF683D4521270F0511A2
dirreq-stats-end 2017-03-01 15:01:19 (86400 s)
dirreq-v3-ips in=8,mx=8,ru=8
dirreq-v3-reqs in=8,mx=8,ru=8
dirreq-v3-resp ok=8,not-enough-sigs=0,unavailable=0,not-found=0,not-modified=0,busy=0
dirreq-v3-direct-dl complete=0,timeout=0,running=0
dirreq-v3-tunneled-dl complete=8,timeout=0,running=0
transport scramblesuit
transport obfs3
transport obfs4
transport fte
bridge-stats-end 2017-03-01 15:03:06 (86400 s)
bridge-ips in=8,ir=8,mx=8,ru=8
bridge-ip-versions v4=8,v6=0
bridge-ip-transports obfs3=8,obfs4=8,scramblesuit=8
router-digest-sha256 50hLT2H4vDO42C/fRWIgV5j3CTldi+ZMPyY3V0IYQSE
router-digest 76BC2C857FDBED685085B16E3852799EF81A7B86
```

A.2.2 Relay extra-info descriptor

```
@type extra-info 1.0
extra-info Pounet27TorRelay EFE68EB2D54E657B5BBF4EB18627646F8DCF66C9
published 2016-12-04 13:01:45
write-history 2016-12-04 10:18:03 (14400 s) 57720832,70514688,199539712,...
read-history 2016-12-04 10:18:03 (14400 s) 64663552,74992640,199556096,498191360,...
dirreq-write-history 2016-12-04 10:18:03 (14400 s) 2048,652288,1426432,937984,...
dirreq-read-history 2016-12-04 10:18:03 (14400 s) 4096,13312,23552,263168,24576,7168
geoip-db-digest C1EB5237F2FBAF63381D8551157F13D12EFCCA25
geoip6-db-digest 1F99B6B0EC78E9DB34D61AE7E0FC261D558E8E5D
dirreq-stats-end 2016-12-03 13:24:35 (86400 s)
dirreq-v3-ips de=8,ua=8
dirreq-v3-reqs de=8,ua=8
dirreq-v3-resp ok=8,not-enough-sigs=0,unavailable=0,not-found=0,not-modified=0,busy=0
dirreq-v3-direct-dl complete=0,timeout=0,running=0
dirreq-v3-tunneled-dl complete=4,timeout=8,running=0
hidserv-stats-end 2016-12-03 18:35:50 (86400 s)
hidserv-rend-relayed-cells 2876020 delta_f=2048 epsilon=0.30 bin_size=1024
hidserv-dir-onions-seen 254 delta_f=8 epsilon=0.30 bin_size=8
entry-stats-end 2016-12-03 18:35:50 (86400 s)
entry-ips us=1064,it=504,fr=472,de=456,es=408,br=224,ru=216,jp=208,pl=192,gb=128,ar=120,th=104,ua=104,nl=88,ca=80,in=80,bg=72,
cell-stats-end 2017-01-30 18:35:50 (86400 s)
cell-processed-cells 5430,23,10,8,7,4,4,3,2,1
cell-queued-cells 0.38,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
cell-time-in-queue 56,0,0,0,0,0,0,0,0,3
cell-circuits-per-decile 15573
conn-bi-direct 2016-12-03 18:35:50 (86400 s) 1417304,46267,48669,100569
router-sig-ed25519 pTwQjRcWzRYJyyhIdfcLia2vhVpn0GgRth7+IpNbyvnATzs5UjQv6v72WSNg8mwg9RzdOpDd+zMQrf5clUnEDA
router-signature
-----BEGIN SIGNATURE-----
M7Ru2Lfaul9AUcmfZ6VFeOkc5kfOmlkQmbescB0aBAYFr0YaC+qbVZKhPEEvNB8d
s6TBjpW5zWmqnDyLNI8klOFtt1Nm0k76Vfb/0Cx5jfiTx0ViyXC0zC0VBG1jmUkX
FxMvXwC049xv2JVXvUupe83xt/13OIgDV0Z8kWYR64g=
-----END SIGNATURE-----
```

References

[1] Roger Dingledine and Nick Mathewson. Tor protocol specification. https://gitweb.torproject.org/torspec.git/. Commit 8eee5024f66d4816d63b341550c01ba4ab059bfc.

[2] Tariq Elahi, George Danezis, and Ian Goldberg. PrivEx: private collection of traffic statistics for anonymous communication networks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 1068–1079, 2014.

[3] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1054–1067. ACM, 2014.

[4] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.

[5] David Goulet, Aaron Johnson, George Kadianakis, and Karsten Loesing. Extrapolating network totals from hidden-service statistics. Technical Report 2015-01-001, The Tor Project, 2015.

[6] Oluwakemi Hambolu. Privacy preserving statistics. Master of Science in Computer Engineering, Clemson University, South Carolina, USA, 2014.

[7] Rob Jansen and Aaron Johnson. Safely measuring tor. In *Proceedings of the 23rd ACM Conference on Computer and Communications Security (CCS '16)*, pages 1553–1567, October 2016.

[8] George Kadianakis and Karsten Loesing. Hidden-service statistics reported by relays. Technical Report 2015-04-001, The Tor Project, 2015.

[9] Karsten Loesing. Analysis of circuit queues in tor. Technical Report 2009-08-001, The Tor Project, 2009.

[10] Karsten Loesing. Counting daily bridge users. Technical Report 2012-10-001, The Tor Project, 2012.

[11] The Tor Project. Metrics web source code. https://gitweb.torproject.org/metrics-web.git/tree. Commit 8bf149b0a89227c56e97a228b2558cacfcecc158.

[12] The Tor Project. Onionoo source code. https://gitweb.torproject.org/onionoo.git/tree. Commit 5b219203b8781b27518133ad7d76e636e82d7fe5.

[13] The Tor Project. Tor source code. https://gitweb.torproject.org/tor.git/tree. Commit a3ce303432f35a6f06f63f0679b9bb577f88dc3c.

[14] The Tor Project. Tor Bugtracker. https://trac.torproject.org/.

[15] Florian Tschorsch and Björn Scheuermann. Distributed privacy-aware user counting. In *HotPETs '11: 4th Workshop on Hot Topics in Privacy Enhancing Technologies*, 2011.

[16] Washington, DC: Central Intelligence Agency. The world factbook 2013-14. https://www.cia.gov/library/publications/the-world-factbook/index.html, 2013. Accessed 2017-04-24.

[4, 3, 6, 15] were recommended by Nick Mathewson (cf. Tor bug tracker ticket #15469, last accessed 2017-04-05).