Overview of Statistical Data in the Tor Network

Karsten Loesing
karsten@torproject.org

Tor Tech Report 2011-03-001*
March 14, 2011

1 Introduction

Statistical analysis in the Tor network can be performed using various kinds of data. In this report we give an overview of three major data sources for statistics in the Tor network: First, we recap measuring the Tor network from public directory information [4] in Section 2 and explain the sanitzation process of (non-public) bridge directory information in Section 3. Second, we describe the numerous aggregate statistics that relays publish about their usage [5] in Sections 4 to 8. Third, we delineate the output of various Tor services like BridgeDB, GetTor, or Tor Check as well as specific measurement tools like Torperf in Sections 11 to 14. All data described in this report are available for download on the metrics website.[1]

2 Server descriptors and network statuses

Relays in the Tor network report their capabilities by publishing server descriptors to the directory authorities. The directory authorities confirm reachability of relays and assign flags to help clients make good path selections. Every hour, the directory authorities publish a network status consensus with all known running relays at the time. Both server descriptors and network statuses constitute a solid data basis for statistical analysis in the Tor network. We described the approach to measure the Tor network from public directory information in [4] and provide interactive graphs on the metrics website.[2] In this section, we briefly describe the most interesting pieces of the two descriptor formats that can be used for statistics.

*This report is superseded by https://metrics.torproject.org/formats.html.
[1]https://metrics.torproject.org/data.html
[2]https://metrics.torproject.org/graphs.html

Server descriptors The server descriptors published by relays at least once every 18 hours contain the necessary information for clients to build circuits using a given relay. These server descriptors can also be useful for statistical analysis of the Tor network infrastructure.

We assume that the majority of server descriptors are correct. But when performing statistical analysis on server descriptors, one has to keep in mind that only a small subset of the information written to server descriptors is confirmed by the trusted directory authorities. In theory, relays can provide false information in their server descriptors, even though the incentive to do so is probably low.

Figure 1 shows an example server descriptor. The following data fields in server descriptors may be relevant to statistical analysis:

- *IP address and ports:* Relays provide their IP address and ports where they accept requests to build circuits and directory requests. These data fields are contained in the first line of a server descriptor starting with `router`. Note that in rare cases, the IP address provided here can be different from the IP address used for exiting to the Internet. The latter can be found in the exit lists produced by Tor Check as described in Section 14.

- *Operating system and Tor software version:* Relays include their operating system and Tor software version in their server descriptors in the `platform` line. While this information is very likely correct in most cases, a few relay operators may try to impede hacking attempts by providing false platform strings.

- *Uptime:* Relays include the number of seconds since the last restart in their server descriptor in the `uptime` line.

- *Own measured bandwidth:* Relays report the bandwidth that they are willing to provide on average and for short periods of time. Relays also perform periodic bandwidth self-tests and report their actual available bandwidth. The latter was used by clients to weight relays in the path selection algorithm and was sometimes subject to manipulation by malicious relays. All three bandwidth values can be found in a server descriptor's `bandwidth` line. With the introduction of bandwidth scanners, the self-reported relay bandwidth in server descriptors has become less relevant.[3]

- *Relay family:* Some relay operators who run more than one relay organize their relays in relay families, so that clients don't pick more than one of these relays for a single circuit. Each relay belonging to a relay family lists the members of that family either by nickname or fingerprint in its server descriptor in the `family` line.

- *Exit policy:* Relays define their exit policy by including firewall-like rules which outgoing connections they reject or accept in the `reject` and `accept` lines.

These are just a subset of the fields in a server descriptor that seem relevant for statistical analysis. For a complete list of fields in server descriptors, see the directory procol specification [1].

[3] http://gitweb.torproject.org/torflow.git/

```
router blutmagie 192.251.226.206 443 0 80
platform Tor 0.2.2.20-alpha on Linux x86_64
opt protocols Link 1 2 Circuit 1
published 2010-12-27 14:35:27
opt fingerprint 6297 B13A 687B 521A 59C6 BD79 188A 2501 EC03 A065
uptime 445412
bandwidth 14336000 18432000 15905178
opt extra-info-digest 5C1D5D6F8B243304079BC15CD96C7FCCB88322D4
opt caches-extra-info
onion-key
[...]
signing-key
[...]
family $66CA87E164F1CFCE8C3BB5C095217A28578B8BAF
  $67EC84376D9C4C467DCE8621AACA109160B5264E
  $7B698D327F1695590408FED95CDEE1565774D136
opt hidden-service-dir
contact abuse@blutmagie.de
reject 0.0.0.0/8:*
reject 169.254.0.0/16:*
reject 127.0.0.0/8:*
reject 192.168.0.0/16:*
reject 10.0.0.0/8:*
reject 172.16.0.0/12:*
reject 192.251.226.206:*
reject *:25
reject *:119
reject *:135-139
reject *:445
reject *:465
reject *:563
reject *:587
reject *:1214
reject *:4661-4666
reject *:6346-6429
reject *:6660-6999
accept *:*
router-signature
[...]
```

Figure 1: Server descriptor published by relay `blutmagie` (without cryptographic keys and hashes)

```
r blutmagie YpexOmh7UhpZxr15GIolAewDoGU
  lFY7WmD/yvVFp9drmZzNeTxZ6dw 2010-12-27 14:35:27 192.251.226.206
  443 80
s Exit Fast Guard HSDir Named Running Stable V2Dir Valid
v Tor 0.2.2.20-alpha
w Bandwidth=30800
p reject 25,119,135-139,445,465,563,587,1214,4661-4666,6346-6429,
  6660-6999
```

Figure 2: Network status entry of relay `blutmagie`

Network statuses Every hour, the directory authorities publish a new network status that contains a list of all running relays. The directory authorities confirm reachability of the contained relays and assign flags based on the relays' characteristics. The entries in a network status reference the last published server descriptor of a relay.

The network statuses are relevant for statistical analysis, because they constitute trusted snapshots of the Tor network. Anyone can publish as many server descriptors as they want, but only the directory authorities can confirm that a relay was running at a given time. Most statistics on the Tor network infrastructure rely on network statuses and possibly combine them with the referenced server descriptors. Figure 2 shows the network status entry referencing the server descriptor from Figure 1. In addition to the reachability information, network statuses contain the following fields that may be relevant for statistical analysis:

- *Relay flags:* The directory authorities assign flags to relays based on their characteristics to the line starting with `s`. Examples are the `Exit` flag if a relay permits exiting to the Internet and the `Guard` flag if a relay is stable enough to be picked as guard node

- *Relay version:* The directory authorities include the version part of the platform string written to server descriptors in the network status in the line starting with `v`.

- *Bandwidth weights:* The network status contains a bandwidth weight for every relay in the lines with `w` that clients shall use for weighting relays in their path selection algorithm. This bandwidth weight is either the self-reported bandwidth of the relay or the bandwidth measured by the bandwidth scanners.

- *Exit policy summary:* Every entry in a network status contains a summary version of a relay's exit policy in the line starting with `p`. This summary is a list of accepted or rejected ports for exit to most IP addresses.

3 Sanitized bridge descriptors

Bridges in the Tor network publish server descriptors to the bridge authority which in turn generates a bridge network status. We cannot, however, make the bridge server descriptors and bridge network statuses available for statistical analysis as we do with the relay server descriptors and relay network statuses. The problem is that bridge server descriptors and network statuses contain bridge IP addresses and other sensitive information that shall not be made publicly available. We therefore sanitize bridge descriptors by removing all potentially identifying information and publish sanitized versions of the descriptors. The processing steps for sanitizing bridge descriptors are as follows:

1. *Replace the bridge identity with its SHA1 value:* Clients can request a bridge's current descriptor by sending its identity string to the bridge authority. This is a feature to make bridges on dynamic IP addresses useful. Therefore, the original identities (and anything that could be used to derive them) need to be removed from the descriptors. The bridge identity is replaced with its SHA1 hash value. The idea is to have a consistent replacement that remains stable over months or even years (without keeping a secret for a keyed hash function).

2. *Remove all cryptographic keys and signatures:* It would be straightforward to learn about the bridge identity from the bridge's public key. Replacing keys by newly generated ones seemed to be unnecessary (and would involve keeping a state over months/years), so that all cryptographic objects have simply been removed.

3. *Replace IP address with IP address hash:* Of course, the IP address needs to be removed, too. It is replaced with `10.x.x.x` with `x.x.x` being the first three bytes of $H(\text{IP address} \,||\, \text{bridge identity} \,||\, \text{secret})$, where $H()$ is SHA-256, $||$ denotes concatenation, and: `IP address` is the 4-byte long binary representation of the bridge's current IP address. `bridge identity` is the 20-byte long binary representation of the bridge's long-term identity fingerprint. `secret` is a 31-byte long secure random string that changes once per month for all descriptors and statuses published in that month.

4. *Replace contact information:* If there is contact information in a descriptor, the contact line is changed to `somebody`.

5. *Replace nickname with Unnamed:* The bridge nicknames might give hints on the location of the bridge if chosen without care; e.g. a bridge nickname might be very similar to the operators' relay nicknames which might be located on adjacent IP addresses. All bridge nicknames are therefore replaced with the string `Unnamed`.

Figure 3 shows an example bridge server descriptor that is referenced from the bridge network status entry in Figure 4. For more details about this process, see the bridge descriptor sanitizer and the metrics database software.[4]

4 Byte histories

Relays include aggregate statistics in their descriptors that they upload to the directory authorities. These aggregate statistics are contained in extra-info descriptors that are published in companion with server descriptors. Extra-info descriptors are not required for clients to build

[4] `https://metrics.torproject.org/tools.html`

```
router Unnamed 10.74.150.129 443 0 0
platform Tor 0.2.2.19-alpha (git-1988927edecce4c7) on Linux i686
opt protocols Link 1 2 Circuit 1
published 2010-12-27 18:55:01
opt fingerprint A5FA 7F38 B02A 415E 72FE 614C 64A1 E5A9 2BA9 9BBD
uptime 2347112
bandwidth 5242880 10485760 1016594
opt extra-info-digest 86E6E9E68707AF586FFD09A36FAC236ADA0D11CC
opt hidden-service-dir
contact somebody
reject *:*
```

Figure 3: Sanitized bridge server descriptor

```
r Unnamed pfp/OLAqQV5y/mFMZKHlqSupm70 dByzfWWLas9cen7PtZ3XGYIJHt4
  2010-12-27 18:55:01 10.74.150.129 443 0
s Fast Guard HSDir Running Stable Valid
```

Figure 4: Sanitized bridge network status entry

circuits. An extra-info descriptor belonging to a server descriptor is referenced by its SHA1 hash value.

Byte histories were the first statistical data that relays published about their usage. Relays report the number of written and read bytes in 15-minute intervals throughout the last 24 hours. The extra-info descriptor in Figure 5 contains the byte histories in the two lines starting with `write-history` and `read-history`. More details about these statistics can be found in the directory protocol specification [1].

5 Directory requests

The directory authorities and directory mirrors report statistical data about processed directory requests. Starting with Tor version 0.2.2.15-alpha, all directories report the number of written and read bytes for answering directory requests. The format is similar to the format of byte histories as described in the previous section. The relevant lines are `dirreq-write-history` and `dirreq-read-history` in Figure 5. These two lines contain the subset of total read and written bytes that the directory mirror spent on responding to any kind of directory request, including network statuses, server descriptors, extra-info descriptors, authority certificates, etc.

The directories further report statistics on answering directory requests for network statuses only. For Tor versions before 0.2.3.x, relay operators had to manually enable these statistics, which is why only a few directories report them. The lines starting with `dirreq-v3-` all belong to the directory request statistics (the lines starting with `dirreq-v2-` report similar statistics for version 2 of the directory protocol which is deprecated at the time of writing this report).

```
extra-info blutmagie 6297B13A687B521A59C6BD79188A2501EC03A065
published 2010-12-27 14:35:27
write-history 2010-12-27 14:34:05 (900 s) 12902389760,
  12902402048,12859373568,12894131200,[...]
read-history 2010-12-27 14:34:05 (900 s) 12770249728,12833485824,
  12661140480,12872439808,[...]
dirreq-write-history 2010-12-27 14:26:13 (900 s) 51731456,
  60808192,56740864,54948864,[...]
dirreq-read-history 2010-12-27 14:26:13 (900 s) 4747264,4767744,
  4511744,4752384,[...]
dirreq-stats-end 2010-12-27 10:51:09 (86400 s)
dirreq-v3-ips us=2000,de=1344,fr=744,kr=712,[...]
dirreq-v2-ips ??=8,au=8,cn=8,cz=8,[...]
dirreq-v3-reqs us=2368,de=1680,kr=1048,fr=800,[...]
dirreq-v2-reqs id=48,??=8,au=8,cn=8,[...]
dirreq-v3-resp ok=12504,not-enough-sigs=0,unavailable=0,
  not-found=0,not-modified=0,busy=128
dirreq-v2-resp ok=64,unavailable=0,not-found=8,not-modified=0,
  busy=8
dirreq-v2-share 1.03%
dirreq-v3-share 1.03%
dirreq-v3-direct-dl complete=316,timeout=4,running=0,min=4649,
  d1=36436,d2=68056,q1=76600,d3=87891,d4=131294,md=173579,
  d6=229695,d7=294528,q3=332053,d8=376301,d9=530252,max=2129698
dirreq-v2-direct-dl complete=16,timeout=52,running=0,min=9769,
  d1=9769,d2=9844,q1=9981,d3=9981,d4=27297,md=33640,d6=60814,
  d7=205884,q3=205884,d8=361137,d9=628256,max=956009
dirreq-v3-tunneled-dl complete=12088,timeout=92,running=4,
  min=534,d1=31351,d2=49166,q1=58490,d3=70774,d4=88192,md=109778,
  d6=152389,d7=203435,q3=246377,d8=323837,d9=559237,max=26601000
dirreq-v2-tunneled-dl complete=0,timeout=0,running=0
entry-stats-end 2010-12-27 10:51:09 (86400 s)
entry-ips de=11024,us=10672,ir=5936,fr=5040,[...]
exit-stats-end 2010-12-27 10:51:09 (86400 s)
exit-kibibytes-written 80=6758009,443=498987,4000=227483,
  5004=1182656,11000=22767,19371=1428809,31551=8212,41500=965584,
  51413=3772428,56424=1912605,other=175227777
exit-kibibytes-read 80=197075167,443=5954607,4000=1660990,
  5004=1808563,11000=1893893,19371=130360,31551=7588414,
  41500=756287,51413=2994144,56424=1646509,other=288412366
exit-streams-opened 80=5095484,443=359256,4000=4508,5004=22288,
  11000=124,19371=24,31551=40,41500=96,51413=16840,56424=28,
  other=1970964
```

Figure 5: Extra-info descriptor published by relay `blutmagie` (without cryptographic signature and with long lines being truncated)

The following fields may be relevant for statistical analysis:

- *Unique IP addresses:* The numbers in `dirreq-v3-ips` denote the unique IP addresses of clients requesting network statuses by country.
- *Network status requests:* The numbers in `dirreq-v3-reqs` constitute the total network status requests by country.
- *Request share:* The percentage in `dirreq-v3-share` is an estimate of the share of directory requests that the reporting relay expects to see in the Tor network. In [2] we found that this estimate isn't very useful for statistical analysis because of the different approaches that clients take to select directory mirrors. The fraction of written directory bytes (`dirreq-write-history`) can be used to derive a better metric for the share of directory requests.
- *Network status responses:* The directories also report whether they could provide the requested network status to clients in `dirreq-v3-resp`. This information was mostly used to diagnose error rates in version 2 of the directory protocol where a lot of directories replied to network status requests with `503 Busy`. In version 3 of the directory protocol, most responses contain the status code `200 OK`.
- *Network status download times:* The line `dirreq-v3-direct-dl` contains statistics on the download of network statuses via the relay's directory port. The line `dirreq-v3-tunnel-ed-dl` contains similar statistics on downloads via a 1-hop circuit between client and directory (which is the common approach in version 3 of the directory protocol). Relays report how many requests have been completed, have timed out, and are still running at the end of a 24-hour time interval as well as the minimum, maximum, median, quartiles, and deciles of download times.

More details about these statistics can be found in the directory protocol specification [1].

6 Connecting clients

Relays can be configured to report per-country statistics on directly connecting clients. This metric includes clients connecting to a relay in order to build circuits and clients creating a 1-hop circuit to request directory information. In practice, the latter number outweighs the former number. The `entry-ips` line in Figure 5 shows the number of unique IP addresses connecting to the relay by country. More details about these statistics can be found in the directory protocol specification [1].

7 Bridge users

Bridges report statistics on connecting bridge clients in their extra-info descriptors. Figure 6 shows a bridge extra-info descriptor with the bridge user statistics in the `bridge-ips` line.

```
extra-info Unnamed A5FA7F38B02A415E72FE614C64A1E5A92BA99BBD
published 2010-12-27 18:55:01
write-history 2010-12-27 18:43:50 (900 s) 151712768,176698368,
  180030464,163150848,[...]
read-history 2010-12-27 18:43:50 (900 s) 148109312,172274688,
  172168192,161094656,[...]
bridge-stats-end 2010-12-27 14:56:29 (86400 s)
bridge-ips sa=48,us=40,de=32,ir=32,[...]
```

Figure 6: Sanitized bridge extra-info descriptor

```
cell-stats-end 2010-12-27 09:59:50 (86400 s)
cell-processed-cells 4563,153,42,15,7,7,6,5,4,2
cell-queued-cells 9.39,0.98,0.09,0.01,0.00,0.00,0.00,0.01,0.00,
  0.01
cell-time-in-queue 2248,807,277,92,49,22,52,55,81,148
cell-circuits-per-decile 7233
```

Figure 7: Cell statistics in extra-info descriptor by relay `gabelmoo`

Bridges running Tor version 0.2.2.3-alpha or earlier report bridge users in a similar line starting with `geoip-client-origins`. The reason for switching to `bridge-ips` was that the measurement interval in `geoip-client-origins` had a variable length, whereas the measurement interval in 0.2.2.4-alpha and later is set to exactly 24 hours. In order to clearly distinguish the new measurement intervals from the old ones, the new keywords have been introduced. More details about these statistics can be found in the directory protocol specification [1].

8 Cell-queue statistics

Relays can be configured to report aggregate statistics on their cell queues. These statistics include average processed cells, average number of queued cells, and average time that cells spend in circuits. Circuits are split into deciles based on the number of processed cells. The statistics are provided for circuit deciles from loudest to quietest circuits. Figure 7 shows the cell statistics contained in an extra-info descriptor by relay `gabelmoo`. An early analysis of cell-queue statistics can be found in [3]. More details about these statistics can be found in the directory protocol specification [1].

9 Exit-port statistics

Exit relays running Tor version 0.2.1.1-alpha or higher can be configured to report aggregate statistics on exiting connections. These relays report the number of opened streams, written and read bytes by exiting port. Until version 0.2.2.19-alpha, relays reported all ports exceeding

```
conn-bi-direct 2010-12-28 15:55:11 (86400 s) 387465,45285,55361,
  81786
```

Figure 8: Bidirectional connection use statistic in extra-info descriptor by relay `zweifaltigkeit`

a threshold of 0.01 % of all written and read exit bytes. Starting with version 0.2.2.20-alpha, relays only report the top 10 ports in exit-port statistics in order not to exceed the maximum extra-info descriptor length of 50 KB. Figure 5 on page 7 contains exit-port statistics in the lines starting with `exit-`. More details about these statistics can be found in the directory protocol specification [1].

10 Bidirectional connection use

Relays running Tor version 0.2.3.1-alpha or higher can be configured to report what fraction of connections is used uni- or bi-directionally. Every 10 seconds, relays determine for every connection whether they read and wrote less than a threshold of 20 KiB. Connections below this threshold are labeled as “Below Threshold”. For the remaining connections, relays report whether they read/wrote at least 10 times as many bytes as they wrote/read. If so, they classify a connection as “Mostly reading” or “Mostly writing,” respectively. All other connections are classified as “Both reading and writing.” After classifying connections, read and write counters are reset for the next 10-second interval. Statistics are aggregated over 24 hours. Figure 8 shows the bidirectional connection use statistics in an extra-info descriptor by relay `zweifaltigkeit`. The four numbers denote the number of connections “Below threshold,” “Mostly reading,” “Mostly writing,” and “Both reading and writing.” More details about these statistics can be found in the directory protocol specification [1].

11 Torperf output files

Torperf is a little tool that measures Tor’s performance as users experience it. Torperf uses a trivial SOCKS client to download files of various sizes over the Tor network and notes how long substeps take. Torperf can be downloaded from the metrics website.[5]

Torperf can produce two output files: `.data` and `.extradata`. The `.data` file contains timestamps for nine substeps and the byte summaries for downloading a test file via Tor. Figure 9 shows an example output of a Torperf run. The timestamps in the upper part of this output are seconds and nanoseconds since 1970-01-01 00:00:00.000000.

Torperf can be configured to write `.extradata` files by attaching a Tor controller and writing certain controller events to disk. The content of a `.extradata` line is shown in the lower part of Figure 9. The first column indicates if this circuit was actually used to fetch the data (`ok`) or if Tor chose a different circuit because this circuit was problematic (`error`). For every `error` entry there should be a following `ok` entry, unless the network of the Torperf instance is dead

[5] `https://metrics.torproject.org/tools.html`

```
# Timestamps and byte summaries contained in .data files:
1293543301 762678   # Connection process started
1293543301 762704   # After socket is created
1293543301 763074   # After socket is connected
1293543301 763190   # After authentication methods are negotiated
                    # (SOCKS 5 only)
1293543301 763816   # After SOCKS request is sent
1293543302 901783   # After SOCKS response is received
1293543302 901818   # After HTTP request is written
1293543304 445732   # After first response is received
1293543305 456664   # After payload is complete
75                  # Written bytes
51442               # Read bytes

# Path information contained in .extradata files:
ok                  # Status code
1293543302          # Circuit build completion time
$2F265B37920BDFE474BF795739978EEFA4427510=fejk4        # 1st hop
$66CA87E164F1CFCE8C3BB5C095217A28578B8BAF=blutmagie3   # 2nd hop
$76997E6557828E8E57F70FDFBD93FB3AA470C620~Amunet8      # 3rd hop
```

Figure 9: Torperf output lines for a single request to download a 50 KiB file (reformatted and annotated with comments)

or the resource is unavailable. The circuit build completion time in the `.extradata` line is the time between Torperf sent a SOCKS request and received a SOCKS response in the `.data` file. The three or more hops of the circuit are listed by relay fingerprint and nickname. An = sign between the two means that a relay has the `Named` flag, whereas the   sign means it doesn't.

12 BridgeDB pool assignment files

BridgeDB is the software that receives bridge network statuses containing the information which bridges are running from the bridge authority, assigns these bridges to persistent distribution rings, and hands them out to bridge users. BridgeDB periodically dumps the list of running bridges with information about the rings, subrings, and file buckets to which they are assigned to a local file. The sanitized versions of these lists containing SHA-1 hashes of bridge fingerprints instead of the original fingerprints are available for statistical analysis.

Figure 10 shows a BridgeDB pool assignment file from March 13, 2011. Every such file begins with a line containing the timestamp when BridgeDB wrote this file. Subsequent lines always start with the SHA-1 hash of a bridge fingerprint, followed by ring, subring, and/or file bucket information. There are currently three distributor ring types in BridgeDB:

1. `unallocated`: These bridges are not distributed by BridgeDB, but are either reserved for manual distribution or are written to file buckets for distribution via an external tool. If a bridge in the `unallocated` ring is assigned to a file bucket, this is noted by `bucket=$bucketname`.

```
bridge-pool-assignment 2011-03-13 14:38:03
00b834117566035736fc6bd4ece950eace8e057a unallocated
00e923e7a8d87d28954fee7503e480f3a03ce4ee email port=443
    flag=stable
0103bb5b00ad3102b2dbafe9ce709a0a7c1060e4 https ring=2 port=443
    flag=stable
[...]
```

Figure 10: BridgeDB pool assignment file from March 13, 2011

```
2010-12-27 - None:167 macosx-i386-bundle:0 macosx-ppc-bundle:0
  source-bundle:2 tor-browser-bundle:0 tor-browser-bundle_ar:0
  tor-browser-bundle_de:0 tor-browser-bundle_en:39
  tor-browser-bundle_es:0 tor-browser-bundle_fa:5
  tor-browser-bundle_fr:0 tor-browser-bundle_it:0
  tor-browser-bundle_nl:0 tor-browser-bundle_pl:0
  tor-browser-bundle_pt:0 tor-browser-bundle_ru:0
  tor-browser-bundle_zh_CN:77 tor-im-browser-bundle:0
  tor-im-browser-bundle_ar:0 tor-im-browser-bundle_de:0
  tor-im-browser-bundle_en:1 tor-im-browser-bundle_es:0
  tor-im-browser-bundle_fa:0 tor-im-browser-bundle_fr:0
  tor-im-browser-bundle_it:0 tor-im-browser-bundle_nl:0
  tor-im-browser-bundle_pl:0 tor-im-browser-bundle_pt:0
  tor-im-browser-bundle_ru:0 tor-im-browser-bundle_zh_CN:0
```

Figure 11: GetTor statistics file for December 27, 2010

2. `email`: These bridges are distributed via an e-mail autoresponder. Bridges can be assigned to subrings by their OR port or relay flag which is defined by `port=$port` and/or `flag=$flag`.

3. `https`: These bridges are distributed via https server. There are multiple https rings to further distribute bridges by IP address ranges, which is denoted by `ring=$ring`. Bridges in the `https` ring can also be assigned to subrings by OR port or relay flag which is defined by `port=$port` and/or `flag=$flag`.

13 GetTor statistics file

GetTor allows users to fetch the Tor software via email. GetTor keeps internal statistics on the number of packages requested every day and writes these statistics to a file. Figure 11 shows the statistics file for December 27, 2010. The `None` entry stands for requests that don't ask for a specific bundle, e.g. requests for the bundle list.

```
ExitNode 63BA28370F543D175173E414D5450590D73E22DC
Published 2010-12-28 07:35:55
LastStatus 2010-12-28 08:10:11
ExitAddress 91.102.152.236 2010-12-28 07:10:30
ExitAddress 91.102.152.227 2010-12-28 10:35:30
```

Figure 12: Exit list entry written on December 28, 2010 at 15:21:44 UTC

14 Tor Check exit lists

TorDNSEL is an implementation of the active testing, DNS-based exit list for Tor exit nodes.[6] Tor Check makes the list of known exits and corresponding exit IP addresses available in a specific format. Figure 12 shows an entry of the exit list written on December 28, 2010 at 15:21:44 UTC. This entry means that the relay with fingerprint `63Ba..` which published a descriptor at 07:35:55 and was contained in a version 2 network status from 08:10:11 uses two different IP addresses for exiting. The first address `91.102.152.236` was found in a test performed at 07:10:30. When looking at the corresponding server descriptor, one finds that this is also the IP address on which the relay accepts connections from inside the Tor network. A second test performed at 10:35:30 reveals that the relay also uses IP address `91.102.152.227` for exiting.

References

[1] Roger Dingledine and Nick Mathewson. Tor directory protocol, version 3. https://gitweb.torproject.org/tor.git/blob_plain/HEAD:/doc/spec/dir-spec.txt.

[2] Sebastian Hahn and Karsten Loesing. Privacy-preserving ways to estimate the number of Tor users. Technical Report 2010-11-001, The Tor Project, November 2010.

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

[4] Karsten Loesing. Measuring the Tor network from public directory information. In *Proc. HotPETS*, Seattle, WA, August 2009. https://metrics.torproject.org/papers/hotpets09.pdf.

[5] Karsten Loesing, Steven J. Murdoch, and Roger Dingledine. A case study on measuring statistical data in the Tor anonymity network. In *Proc. Workshop on Ethics in Computer Security Research*, Tenerife, Canary Islands, Spain, January 2010. https://metrics.torproject.org/papers/wecsr10.pdf.

[6] https://www.torproject.org/tordnsel/dist/