Censored Planet Observatory

Release 2.0

Feb 20, 2022
1 Contents

1.1 Censored Planet Version Timeline ................................................................. 3
1.2 DNS Data - Satellite ......................................................................................... 3
  1.2.1 Satellite-v1 ............................................................................................... 4
    1.2.1.1 Satellite-v1.0 .................................................................................... 5
  1.2.2 Satellite-v2 ............................................................................................... 8
    1.2.2.1 Satellite-v2.0 .................................................................................... 9
    1.2.2.2 Satellite-v2.1 .................................................................................. 13
    1.2.2.3 Satellite-v2.2 .................................................................................. 13
  1.2.3 Notes ......................................................................................................... 16
1.3 HTTP(S) Data - Hyperquack .......................................................................... 16
  1.3.1 Hyperquack-v1 ........................................................................................ 17
    1.3.1.1 Data Format ...................................................................................... 18
  1.3.2 Hyperquack-v2 ........................................................................................ 19
    1.3.2.1 Data Format ...................................................................................... 20
  1.3.3 Notes ......................................................................................................... 22
1.4 Censored Planet Data Analysis ....................................................................... 22
  1.4.1 Censored Planet Data Analysis Pipeline ................................................... 22
  1.4.2 Censored Planet Dashboard ...................................................................... 23
    1.4.2.1 FAQ ................................................................................................. 24
  1.4.3 Censored Planet Data Analysis Tool ........................................................ 27

2 Indices and tables ........................................................................................... 29
Censored Planet is a longitudinal censorship measurement platform that collects remote measurement measurements in more than 200 countries. Censored Planet was launched in August 2018, and has since then collected more than 45 billion measurement data points. Censored Planet measures network interference on the TCP/IP, DNS, and HTTP(S) protocols, using remote measurement techniques Augur, Satellite, and Hyperquack respectively. Every week, Censored Planet collects reachability data about 2000 popular and sensitive websites from more than 95,000 vantage points around the world. An academic paper about Censored Planet can be found here.

Censored Planet’s measurement data has been crucial in identifying and monitoring several important censorship and network interference events. In 2019, Censored Planet data was used to study the large-scale HTTPS interception that occurred in Kazakhstan, and was instrumental in driving changes in major web browsers that blocked the interception attack. Censored Planet data has been used to study Russia’s decentralized censorship mechanism, and the throttling attack they performed on Twitter. Censored Planet has also been used to identify the deployment of network censorship devices, and track the blocking of COVID-19 related websites around the world.

Censored Planet data is available to the public through the Censored Planet website: data.censoredplanet.org/raw. The Censored Planet raw data website contains archived compressed data files corresponding to one scan using each measurement technique.

Figure - Raw data files on the Censored Planet website

Each scan contains a test of all 2,000 websites tested by Censored Planet with a set of vantage points across different countries. The archive files can be downloaded, extracted and analyzed. The data formats change based on the version of the measurement technique. The data formats and tips for analyzing the data for each of the published data files and versions are available below. It is critical that the raw data produced by Censored Planet is treated as a network observation and analyzed further using our analysis pipeline before use. For more information about using the data, please refer to the Censored Planet Github, or email Censored Planet at censoredplanet@umich.edu.
1.1 Censored Planet Version Timeline

The timelines of the data generated by different versions of Censored Planet’s remote measurement techniques are shown below. Refer to documentation about each of the techniques to get information about the data format for each version.

<table>
<thead>
<tr>
<th>Satellite</th>
<th>Hyperquack</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1</td>
<td>&gt;= 2018-07-27 &amp;&amp; &lt;= 2021-04-22</td>
</tr>
<tr>
<td>v2.0</td>
<td>&gt;= 2021-04-25 &amp;&amp; &lt;= 2021-05-30</td>
</tr>
<tr>
<td>v2.1</td>
<td>&gt;= 2021-06-01 &amp;&amp; &lt;= 2021-07-21</td>
</tr>
<tr>
<td>v2.2</td>
<td>&gt;= 2021-07-21</td>
</tr>
</tbody>
</table>

1.2 DNS Data - Satellite

Satellite/Iris is Censored Planet's remote measurement technique that detects DNS interference using Open DNS resolvers. Below, we provide an overview of Satellite and its data format. Refer to our academic papers for in-depth details about Satellite.
1.2.1 Satellite-v1

Satellite-v1 is the first version of Satellite that we operated from August 2018 - February 2021. The primary function of Satellite is to detect incorrect DNS resolutions from open DNS resolvers in many countries.

- From a measurement machine at the University of Michigan, we send a DNS query for a website whose reachability we're interested in, to an open DNS resolver in a country of interest (1). The response from the DNS resolver is our Test IP (2).
- We also send a DNS query for the same website to trusted control resolvers (3), and record their response as the control IP (4).
- We then compare the test and control responses using several heuristics, including a direct IP address comparison, and comparison of the AS number, AS names, HTTP content hashes, and TLS certificates associated with the test and control IP addresses (5). Satellite-v1 only labels a measurement as an anomaly when all of the heuristics mismatch.

Our various publications and reports have used Satellite-v1 to detect many cases of DNS manipulation. For instance, in our recent investigation into the filtering of COVID-19 websites, Satellite-v1 found many networks using website filtering products to manipulate DNS responses of COVID-related websites.
1.2.1.1 Satellite-v1.0

Data Format

The published data has the following directory structure:

```
CP_Satellite-YYYY-MM-DD-HH-MM-SS/
|-- answers_control.json
|-- answers_err.json
|-- answers_ip.json
|-- answers.json
|-- answers_raw.json
|-- dns.pkt
|-- interference_err.json
|-- interference.json
|-- resolvers_err.json
|-- resolvers_ip.json
|-- resolvers.json
|-- resolvers_ptr.json
|-- resolvers_raw.json
|-- tagged_answers.json
|-- tagged_resolvers.json
```

Output

The relevant output is located in the `raw/` directory.

Probe

1. Generate a DNS A query packet for a controlled domain (`dns.pkt`).
2. Perform a ZMap scan with the probe packet for open DNS resolvers. `resolvers_raw.json` contains the ZMap output:
   ```
   • `saddr` [String] IP address of a DNS resolver.
   • `data` [String] Raw response to probe domain.
   ```

Filter

1. Check the probe responses of the resolvers found by ZMap. `resolvers_ip.json` contains resolvers that returned the correct probe response:
   ```
   • `resolver` [String] IP address of a DNS resolver.
   • `answer` [String] The resolver’s response (IP address) to the probe domain.
   ```
2. Perform PTR queries on the IPs of resolvers with the correct probe response. `resolvers_err.json` contains resolvers with failed PTR queries:
   ```
   • `resolver` [String] IP address of a DNS resolver.
   • `error` [JSON Object] Contains error information.
   ```
resolvers_ptr.json contains resolvers with successful PTR queries:

- **resolver** [String] IP address of a DNS resolver.
- **names** [Array] Result from PTR query (the hostname).

3. Identify infrastructure resolvers from successful PTR queries and add predefined “control” and “special” resolvers to form final set of vantage points.

resolvers.json contains the infrastructure, “control”, and “special” resolvers.

- **resolver** [String] IP address of a DNS resolver.
- **name** [String] Result from PTR query (if infrastructure), “control”, or “special”.

Query

1. Make DNS queries for each test domain to each resolver.

answers_err.json contains erroneous queries:

- **resolver** [String] The IP address of the vantage point (a DNS resolver).
- **query** [String] The domain being queried.
- **error** [String / JSON Object] Either “no_answer” or a dictionary with additional error information.

Note:

- In some cases, the resolver field may be replaced by ip - both are referring to the resolver’s IP.
- “no_answer” appears in the error field if no A resource records (IPs) are returned - this includes the NXDOMAIN response.
- Responses with NXDOMAIN or other errors may indicate censorship. However, these cases are not analyzed further in Satellite-v1.

answers_raw.json contains raw responses from successful queries:

- **resolver** [String] The IP address of the vantage point (a DNS resolver).
- **query** [String] The domain being queried.
- **data** [String] Raw query response.

2. Separate responses (converted to IP addresses) from control resolvers and non-control resolvers.

answers_control.json contains responses for queries to control resolvers:

- **resolver** [String] The IP address of the vantage point (a DNS resolver).
- **query** [String] The domain being queried.
- **answers** [Array] The resolver’s response for the queried domain (list of answer IPs).

answers.json contains responses for queries to non-control resolvers:

- **resolver** [String] The IP address of the vantage point (a DNS resolver).
- **query** [String] The domain being queried.
- **answers** [Array] The resolver’s response for the queried domain (list of answer IPs).

3. Determine set of IP addresses that appeared across all query responses for tagging.

answers_ip.json contains these IPs, one IP per line:
Tag

1. Tag each answer IP with information from Censys.

   tagged_answers.json contains the answer IPs and their HTTP, TLS, and AS tags:
   - **ip** [String] An IP address from a query response.
   - **http** [String] The hash of the HTTP body.
   - **cert** [String] The hash of the TLS certificate.
   - **asname** [String] The autonomous system (AS) name.
   - **asnum** [Integer] The autonomous system (AS) number.

   Note:
   - Fields may have null values if the information was not available on Censys.

2. Tag each resolver with the location from Maxmind.

   tagged_resolvers.json contains the resolvers and their countries:
   - **resolver** [String] The IP address of the vantage point (a DNS resolver).
   - **country** [String] The full name of the country where the resolver is located.

Detect

1. Compare query responses between non-control resolvers and control resolvers to identify interference.

   interference_err.json contains resolver responses for queries with no control response:
   - **resolver** [String] The IP address of the vantage point (a DNS resolver).
   - **query** [String] The domain being queried.
   - **answers** [Array] The resolver’s response for the queried domain (list of answer IPs).

   interference.json contains the interference assessment for the remaining resolver responses:
   - **resolver** [String] The IP address of the vantage point (a DNS resolver).
   - **query** [String] The domain being queried.
   - **answers** [JSON object] The resolver’s returned answer IPs for the queried domain are the keys. Each answer IP is mapped to an array of its tags that matched the control tags - if the IP is in the control set, “ip” is appended and if the IP has no tags, “no_tags” is appended.
   - **passed** [Boolean] Equals true if interference is not detected. Note that if this field is set to false, it may indicate either DNS interference, or an unexpected answer for the resolution. Further manual confirmation is required to confirm censorship.

   Note:
   - For each response, the answer IPs and their tags are compared to the set of answer IPs and tags from all the control resolvers for the same query domain. A response is classified as interference if there is no overlap between the two.
   - Cases where the control answer IPs have no tags will be considered interference if the resolver’s answer IPs are not in the control set.
• Satellite-v1 anomalies (interference detected) need to be explicitly confirmed by fetching pages hosted at the resolved IPs in post-processing. This functionality is included by default in Satellite v2.

Limitations

Although Satellite-v1 was extremely useful in detecting DNS interference at large scale, it suffered from several limitations, which form the improvements in Satellite-v2.x.

• Satellite-v1 could not detect DNS censorship where A records were not available i.e. Satellite-v1 primarily focused on detecting incorrect DNS resolutions through the resolved IP address, and did not contain heuristics to measure DNS manipulation which manifested through timeouts, NXDOMAIN responses, SERVFAIL responses, etc.

• Satellite-v1 required post-processing to remove false positives and confirm the presence of anomalies, such as through using post-measurement heuristics and blockpage regexes. Satellite-v2 has the inbuilt capability to perform most post-processing measurements.

• Satellite-v1’s data format provided results in several files, which were hard to parse. With Satellite-v2, our aim is also to present an easier, more intuitive data format.

1.2.2 Satellite-v2

Figure - Overview of Satellite-v2

Satellite-v2 is our brand new version of Satellite, where we’ve made several modifications to the measurement technique and data format for facilitating accurate and efficient remote DNS interference measurements. Below, we detail the major changes we’ve made in Satellite-v2.
• **Measuring DNS interference without A records** - In Satellite-v2, we have added a sandwiched retry mechanism to our Satellite measurements in order to detect DNS interference that results in a non-zero R code response. A description of the method is shown in the figure below. We first make a control query to the open DNS resolver, providing a domain name that we do not expect to be blocked (eg. www.example.com). After the control query, we make up to 4 retries of the test DNS query, providing the test domain name. In case an A record is detected, we stop the test measurement. At the end, we perform another control query similar to the first measurement. The control queries ensure that the resolver is behaving correctly for an innocuous domain, and the multiple retry mechanism accounts for temporary errors in the network. With the help of the sandwiched retry mechanism, Satellite-v2 is able to detect DNS interference that manifests as timeouts, NXDOMAIN, SERVFAIL etc. From our preliminary analysis of Satellite-v2 data, we’ve already found several cases of DNS interference that can be identified using this method. For example, from the Satellite-v2 scan performed on 2021-03-17, we are able to identify 174,795 responses that have non-zero R codes from China, which makes up 15.6% out of the responses marked as interference. This kind of DNS interference was previously omitted by satellite v1. Shown below is an example measurement that passed the sandwich control tests, but received server failure R code. This could be an indicator of censorship or geoblocking.

• **Fetching HTML pages hosted at resolved IPs marked as an anomaly** - Satellite-v2 has an in-built fetch feature that performs HTTP and HTTPS GET requests to resolved IPs that fail our heuristics, and we store the HTML responses in blockpages.json. Satellite-v2 data files available on our website contain this file for easier confirmation of DNS censorship, while this step was being performed as a post-processing step in Satellite-v1. This addition helps in quickly identifying blockpages such as the example shown in the figure below.

• **Adding scan-level heuristics to exclude false positives** - Another step part of the post-processing pipeline of Satellite-v1 that is inbuilt in Satellite-v2. We exclude potentially false positive anomalies by using scan-level heuristics, such as the number of domains resolving to the anomalous IP address, or the anomalous IP address being part of a big CDN. Note that this step may lead to Satellite-v2 missing certain censorship. This output can be found in results_verified.json.

• **Other changes** - We updated the heuristics to determine whether a DNS response is interfered - Satellite-v2 now includes a new “confidence” field, which addresses the certainty of interference according to the state of comparison between responses from the test resolvers and the control resolvers. We also make sure that IPs with no metadata information from Censys are not marked as interference.

We have also reorganized our output files so that they are easier to read. The primary output files containing DNS interference data are results.json and results_verified.json. Satellite-v2 integrates more information in the results.json file, like the country name and country code of the target resolver, and start time and end time of each measurement. We hope this modification makes processing of the satellite data easier for our users.

### 1.2.2.1 Satellite-v2.0

**Data Format**

The published data has the following directory structure:

```
CP_Satellite-YYYY-MM-DD-HH-MM-SS/
|   |-- log.json
|   |-- raw/
|   |   |-- blockpages.json
|   |   |-- dns.pkt
|   |   |-- resolvers_err.json
|   |   |-- resolvers_ip.json
|   |   |-- resolvers.json
|   |   |-- resolvers_ptr.json
|   |   |-- resolvers_raw.json
```

(continues on next page)
1. Generate a DNS A query packet for a controlled domain (*dns.pkt*).

2. Perform a ZMap (Internet-wide) scan with the probe packet for open DNS resolvers.

   resolvers_raw.json contains the ZMap output:
   - `saddr` [String] IP address of a DNS resolver.
   - `data` [String] Raw response to probe domain.

1. Check the probe responses of the resolvers found by ZMap.

   resolvers_ip.json contains resolvers that returned the correct probe response:
   - `vp` [String] The IP address of the potential vantage point (a DNS resolver).
   - `response` [String] The resolver’s response (IP address) to the probe domain.

2. Perform PTR queries on the IPs of resolvers with the correct probe response.

   resolvers_ptr.json contains resolvers with successful PTR queries:
   - `vp` [String] The IP address of the potential vantage point (a DNS resolver).
   - `names` [Array] Result from PTR query (the hostname).

3. Identify infrastructure resolvers from successful PTR queries and add predefined “control” and “special” resolvers to form final set of vantage points.

   resolvers.json contains the infrastructure, “control”, and “special” resolvers.
   - `vp` [String] The IP address of the vantage point (a DNS resolver).
   - `name` [String] Result from PTR query (if infrastructure), “control”, or “special”.

1. Make DNS queries for each test domain to each resolver.

   responses_raw.json contains raw responses from successful queries:
   - `vp` [String] The IP address of the vantage point (a DNS resolver).
   - `test_url` [String] The test domain being queried.

**Note:**
- **NEW:** The query for the test domain is attempted up to four times in case of non Type A response.
  To check the status of the resolver, a control domain is queried before and after the queries for the test domain.
2. Parse and separate responses from control resolvers and non-control resolvers.

`responses_control.json` contains responses for queries to control resolvers and `responses.json` contains responses for queries to non-control resolvers:

- **vp** [String] The IP address of the vantage point (a DNS resolver).
- **test_url** [String] The test domain being queried.
- **response** [Array] The resolver’s responses for the control and test domain - in the order control domain, test domain (up to 4 attempts), control domain.
  - **url** [String] The domain being queried in this trial (either the control domain or test_url)
  - **has_type_a** [Boolean] Equals true if the query returned a valid A resource record.
  - **answer** [Array] The resolver’s response for the queried domain in this trial (list of answer IPs if successful).
  - **error** [String] Contains error information.
  - **rcode** [Integer] Response code mapping to success (0) or errors (>0).
  - **start_time** [String] The start time of the measurement.
  - **end_time** [String] The end time of the measurement.
- **resolver_status** [Boolean] Equals true if the resolver successfully responds to the two control queries.
- **raw** [Array] The resolver’s unparsed responses (corresponding to the respective index in response).

3. Determine set of IP addresses that appeared across all query responses for tagging.

`responses_ip.json` contains these IPs, one IP per line:

- **response** [String] An IP address from a query response.

1. Tag each answer IP with information from Censys.

`tagged_responses.json` contains the answer IPs and their HTTP, TLS, and AS tags:

- **ip** [String] An IP address from a query response.
- **http** [String] The hash of the HTTP body.
- **cert** [String] The hash of the TLS certificate.
- **asname** [String] The autonomous system (AS) name.
- **asnum** [Integer] The autonomous system (AS) number.

**Note:**
- Fields may have null values if the information was not available on Censys.

2. Tag each resolver with the location from Maxmind.

`tagged_resolvers.json` contains the resolvers and their countries:

- **vp** [String] The IP address of the vantage point (a DNS resolver).
- **location** [JSON object]
  - **country_name** [String] The full name of the country where the resolver is located.
- **country_code** [String] The two-letter ISO 3166 code of the country where the resolver is located.

1. Compare query responses between non-control resolvers and control resolvers to identify interference.

   **results.json** contains the interference assessment for the query responses:
   
   - **vp** [String] The IP address of the vantage point (a DNS resolver).
   - **location** JSON object
     - **country_name** [String] The full name of the country where the resolver is located.
     - **country_code** [String] The two-letter ISO 3166 code of the country where the resolver is located.
   - **test_url** [String] The domain being queried.
   - **response** [JSON object] The resolver’s returned answer IPs for the queried domain are the keys. Each answer IP is mapped to an array of its tags that matched the control tags - if the IP is in the control set, “ip” is appended and if the IP has no tags, “no_tags” is appended. Also has an **rcode** field mapping to a list of response codes for the trials.
   - **passed_control** [Boolean] Equals true if both control queries were successful.
   - **in_control_group** [Boolean] Equals true if at least one control resolver had a valid response for this test domain.
   - **connect_error** [Boolean] Equals true if all test domain query attempts returned errors.
   - **anomaly** [Boolean] Equals true if an anomaly is detected. In case there are no tags for the answers or control, then this field is conservatively marked as false.
   - **start_time** [String] The start time of the measurement.
   - **end_time** [String] The end time of the measurement.
   - **confidence** [JSON object]
     - **average** [Float] Average percentage of tags matching the control set for the answers (average of matches).
     - **matches** [Array] Contains the percentage of tags matching the control set for each answer. If an answer IP is in the control set, the percentage for that answer is 100 even if the IP has no tags.
     - **untagged_controls** [Boolean] Equals true if all control IPs for the query have no tags.
     - **untagged_answers** [Boolean] Equals true if all answer IPs have no tags.

**Note:**

- For each response, the answer IPs and their tags are compared to the set of answer IPs and tags from all the control resolvers for the same query domain. A response is classified as an anomaly if there is no overlap between the two.

1. Perform HTTP(S) GET requests to the IPs identified as anomalies.

   **blockpages.json** contains the responses:
   
   - **ip** [String] The IP address from an anomalous DNS response.
   - **keyword** [String] The domain queried for the anomalous DNS response.
   - **http** [Object] HTTP response.
   - **https** [Object] HTTPS response.
1. New heuristics to exclude possible cases of erroneous answers from resolvers. Currently, verify excludes answer IPs that are part of big CDNs (Note: this could lead to false negatives). results_verified.json contains only the rows that were earlier marked as anomalies:

- **vp** [String] The IP address of the vantage point (a DNS resolver).
- **location:** JSON object
  - **country_name** [String] The full name of the country where the resolver is located.
  - **country_code** [String] The two-letter ISO 3166 code of the country where the resolver is located.
- **test_url** [String] The domain being queried.
- **response** [JSON object] The resolver’s returned answer IPs for the queried domain are the keys. Each answer IP is mapped to an array of its tags that matched the control tags - if the IP is in the control set, “ip” is appended and if the IP has no tags, “no_tags” is appended. Also has an rcode field mapping to a list of response codes for the trials.
- **excluded** [Boolean] Should this observation be excluded from being counted as an anomaly?
- **exclude_reason** [String Array] If observation should be excluded, why? (eg. “is_CDN”)

### 1.2.2.2 Satellite-v2.1

Satellite-v2.1 incorporates minor changes from Satellite-v2.0, starting after April 14, 2021. The changes include, * In the filter module, we removed resolvers from resolvers.json if they can not resolve the root server * We removed the liveness test response from results.json and results_verified.json. * We removed the error messages from results.json and results_verified.json, if any.

### 1.2.2.3 Satellite-v2.2

Satellite-v2.2 incorporates major changes in code and data structure from Satellite-v2.1, but no major changes in the functionality of Satellite. The changes are made after June 7, 2021 and they include, * Store information generated from the query, tag, detect, and verify module in memory, producing only one file (results.json) as output, instead of generating outputs for every module. Renamed query-tag-detect-verify as “test” module, and probe-filter as “discovery”. * Updated test module so that it first conducts queries for control resolvers, and then query, tag and detect test resolvers in batches.

### Data Format

The published data has the following directory structure:

```bash
CP_Satellite-YYYY-MM-DD-HH-MM-SS/
|-- resolvers_raw.json
|-- dns.pkt
|-- resolvers.json
|-- results_verified.json
|-- blockpages.json
```

Satellite v2 is divided into three parts:
1. **discovery**: consist of probe and filter modules.

2. **test**: consist of query, tag and detect modules.

3. verification and blockpage fetching: consist of fetch and verify.

1. Generate a DNS A query packet for a controlled domain (`dns.pkt`).
   
   2. Perform a ZMap (Internet-wide) scan with the probe packet for open DNS resolvers.

   resolvers_raw.json contains the ZMap output:
   - `saddr` [String] IP address of a DNS resolver.
   - `data` [String] Raw response to probe domain.

1. Perform PTR queries on the IPs of resolvers found by ZMap and filter out the ones without PTR records.

2. Perform Liveness test on the infrastructural resolvers and filter out the ones that fail.

3. Add predefined “control” and “special” resolvers to form the final set of vantage points.

4. Tag each resolver with the location from Maxmind.

   resolvers.json contains the infrastructure, “control”, and “special” resolvers.
   - `vp` [String] The IP address of the vantage point (a DNS resolver).
   - `name` [String] Result from PTR query (if infrastructure), “control”, or “special”.
     - `location` JSON object
       - `country_name` [String] The full name of the country where the resolver is located.
       - `country_code` [String] The two-letter ISO 3166 code of the country where the resolver is located.

1. Make DNS queries for each test domain to each resolver. The query for the test domain is attempted up to four times in case of connection error. To check the status of the resolver, a control measurement is conducted before the queries for the test domain. If the first control measurement fails, no further measurements will be conducted for the same (resolver, domain) pair. If all 4 trials for the test domain fail, another control measurement will be conducted.

2. Parse and separate responses from control resolvers and non-control resolvers.

   1. **Tag each answer IP with information from Censys. Note:**
      - Fields may have empty strings if the information was not available on Censys.

1. Compare query responses between non-control resolvers and control resolvers to identify interference. When running satellite v2 as a whole module, detect does not output any files. However, when run separately, detect outputs results.json with the excluded field set to `false` and the excluded_reason field set to `null` by default. (See the output structure in verify section)

   Note:
   - For each response, the answer IPs and their tags are compared to the set of answer IPs and tags from all the control resolvers for the same query domain. A response is classified as an anomaly if there is no overlap between the two.

1. Perform HTTP(S) GET requests to the IPs identified as anomalies.

   blockpages.json contains the responses:
   - `ip` [String] The IP address from an anomalous DNS response.
   - `keyword` [String] The domain queried for the anomalous DNS response.
• **http** [Object] HTTP response.
• **https** [Object] HTTPS response.
• **fetched** [Boolean] Equals true if a page is successfully fetched.
• **start_time** [String] The start time of the measurement.
• **end_time** [String] The end time of the measurement.

1. New heuristics to exclude possible cases of erroneous answers from resolvers. Currently, verify excludes answer IPs that results_verified.json contains all the information when running full mode.
   • **vp** [String] The IP address of the vantage point (a DNS resolver).
   • **test_url** [String] The test domain being queried.
   • **location**: JSON object
     - **country_name** [String] The full name of the country where the resolver is located.
     - **country_code** [String] The two-letter ISO 3166 code of the country where the resolver is located.
   • **passed_liveness** [Boolean] Equals false if both control queries were unsuccessful.
   • **in_control_group** [Boolean] Equals true if at least one control resolver had a valid response for this test domain.
   • **connect_error** [Boolean] Equals true if all test domain query attempts returned errors. This field is also set to be true if the first control measurement fails, and no further measurements for the test domain are conducted. Use this field in conjunction with the passed_liveness field to find anomalies.
   • **anomaly** [Boolean] Equals true if an anomaly is detected. In case there are no tags for the answers or control, then this field is conservatively marked as false.
   • **start_time** [String] The start time of the measurement.
   • **end_time** [String] The end time of the measurement.
   • **response**: JSON object
     The resolver’s returned answers for the queried domain are the keys.
     - **url**: String The domain being queried in this trial, either the control domain for liveness test or test_url. The liveness test DNS responses are only recorded if they do not contain a type-A RR.
     - **has_type_a**: Boolean Equals true if the query returned a valid A resource record.
     - **error**: String Contains error information.
     - **rcode**: Integer Response code mapping to success (0) or errors (-1 for connection error, > 0 for errors specified in RFC 2929).
     - **response**: JSON Object Consist of a map between IPs the resolver responded for the queried domain and tags from Maxmind:
       * **http** [String] The hash of the HTTP body.
       * **cert** [String] The hash of the TLS certificate.
       * **asname** [String] The autonomous system (AS) name.
       * **asnum** [Integer] The autonomous system (AS) number.
matched [Array] An array of its tags that matched the control tags - if the IP is in the control set, “ip” is appended and if the IP has no tags, “no_tags” is appended.

- confidence [JSON object]
  - average [Float] The average percentage of tags matching the control set for the answers (average of matches).
  - matches [Array] Contains the percentage of tags matching the control set for each answer. If an answer IP is in the control set, the percentage for that answer is 100 even if the IP has no tags.
  - untagged_controls [Boolean] Equals true if all control IPs for the query have no tags.
  - untagged_answers [Boolean] Equals true if all answer IPs have no tags.

- excluded [Boolean] Equals true if this observation should be excluded from being counted as an anomaly.

- exclude_reason [String Array] The reasons that this observation should be excluded (eg. “is_CDN”)

### 1.2.3 Notes

While Satellite includes multiple control resolvers intended to avoid false inferences there is still a possibility that certain measurements are marked as anomalies incorrectly. To confirm censorship, it is critical that the raw DNS responses are compared to known blockpage fingerprints. The blockpage fingerprints currently recorded by Censored Planet are available here. Moreover, aggregations can be used to avoid anomalous vantage points and domains. Please use our analysis pipeline to process the data before using it.

Censored Planet detects network interference of websites using remote measurements to infrastructural vantage points within networks (eg. institutions). Note that this raw data cannot determine the entity responsible for the blocking or the intent behind it. Please exercise caution when using the data, and reach out to us at censoredplanet@umich.edu if you have any questions.

### 1.3 HTTP(S) Data - Hyperquack

Hyperquack (and Quack) is Censored Planet’s measurement techniques that measure application-layer interference using the Echo, Discard, HTTP, and HTTPS protocols. Below, we provide a detailed overview of Hyperquack, and the data formats of Hyperquack data published on the Censored Planet website. Refer to our academic papers for more information about Quack and Hyperquack.
1.3.1 Hyperquack-v1

From a remote measurement machine, we send an HTTP get look-alike request containing a non-sensitive control URL to a vantage point’s Echo or Discard port. Vantage points are selected from infrastructural servers such as ISP routers to minimize risk to their owners. We observe the result, and if the port is responding incorrectly according to its protocol, we abort the test, mark the vantage point as broken, and remove the vantage point from our test list.

If the control test succeeds, we then send an HTTP get look-alike request containing a potentially sensitive URL to the vantage point. If the vantage point responds correctly, we record that there is not an anomaly. If the vantage point responds incorrectly, we repeat the request up to four more times. If any such request results in a correct response, we again record that there is not an anomaly.

If all five requests result in incorrect responses, we then send another request containing a control keyword. If this request results in a correct response, we record the possibility of interference.

If this control request results in an incorrect response, we wait some time then resend the request, to account for stateful interference. If the second request fails, we mark the vantage point as broken and remove the vantage point from our test list. If the request results in a correct response, we mark both potential interference and stateful interference.

Hyperquack-v1 is built up from the Quack-v1 protocol to include support for the HTTP and HTTPS protocols. Before performing any tests, we send multiple HTTP get requests containing non-sensitive control URLs to each of the vantage points we are testing. If the responses to all of the requests are consistent, the responses are stripped of dynamic content such as cookies and turned into a template for the vantage point. Then when performing the tests with the sensitive keywords, we compare the vantage point’s response to its template.

Our various publications and reports have used Quack-v1 and Hyperquack-v1 to detect many cases of application-layer interference. For instance, in our recent investigation into the filtering of COVID-19 websites <https://censoredplanet.org/assets/covid.pdf>, Quack-v1 was used to detect censorship in unexpected places like Canada.
### Data Format

The published data has the following directory structure:

<table>
<thead>
<tr>
<th>CP_Quack-PROTOCOL-YYYY-MM-DD-HH-MM-SS/</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

### Outputs

The relevant output is located in `results.json`.

### Fields

- **Server** [String] The IP address of the vantage point used in this trial.
- **Keyword** [String] The URL being tested by this trial.
- **Retries** [Integer] The number of times Hyperquack had to resend the test packet in the course of the trial. For example, if the first packet sent to the vantage point returned the expected response, this field will be set to 0. If the first packet does not yield the expected response, every other packet sent by the system will increment this field by 1.
- **Results** An array representing the results of each probe sent to the vantage point. Each entry is a JSON object with six subfields:
  - **Sent** [String] The contents of the packet sent to the vantage point. Note that this field may not be populated in cases where there is no application-layer response from the vantage point. In case the TCP handshake with the vantage point fails, the error field will be set accordingly.
  - **Received** [JSON Object] If the response given by the vantage point does not match the template, Hyperquack-v1 will add this field. Describes the response sent by the vantage point, including HTTP headers, the HTTP response code, the body of the response, and any TLS information.
  - **Success** [Boolean] Each trial performed by Hyperquack determines whether or not the probe was interfered with by comparing the response returned by the vantage point to an already known template. If the response does not match, that is potentially evidence of interference. Set to `true` if the response given by the vantage point matches the known template, and `false` otherwise.
  - **Error** [String] If the probe fails with an error, that is potential evidence of blocking. If this occurs, this field will be populated. Describes the encountered error. Note that this field can be used to filter out TCP handshake and setup errors.
  - **StartTime** [Timestamp] The time when the probe was sent.
  - **EndTime** [Timestamp] The time when the response to the probe response arrived.
- **Blocked** [Boolean] Indicates whether the probes to the vantage point show enough evidence to conclude that the vantage point has observed some sort of anomaly, potentially indicative of blocking.
- **FailSanity** [Boolean] Set to `true` when all control probes sent to the vantage point fail to match the known template. This implies that the mismatching responses are due to an error in the vantage point or the network, not censorship. Rows with FailSanity set to `true` should not be considered for analysis.
1.3.2 Hyperquack-v2

Hyperquack-v2 is our new version of both the Quack and Hyperquack measurement techniques. We’ve restructured the system to work as a request-based measurement server rather than a single-use measurement program. A user will run the program on a machine that will act as a server, and then users can interact with the program using a JSON API. The implications of this restructure are as follows.

- **Flexibility in Scheduling** - Unlike in Quack-v1 and Hyperquack-v1, when a scan is performed using Hyperquack-v2, a list of vantage points are added to Hyperquack-v2, then test keywords are added as work for the server to complete. When adding work, the user can specify which vantage points that work applies to, such as specifying all the vantage points in a given country, all the vantage points in a given subnet, or simply a list of specific vantage points. This allows users to more easily schedule targeted scans. To make differentiating between these concurrent scans easier, we also added a tagging system that allows for the output of Hyperquack-v1 to be redirected to custom files.

- **On-the-fly Changes to Scans** - As a scan is running, the user can call endpoints to add work, add more vantage points, or remove vantage points. This further increases the flexibility of Hyperquack-v2, as scans can be updated in the middle of running as opposed to being re-run with updated parameters in Quack-v1 and Hyperquack-v1.

- **Stronger Vantage Point Evaluation** - In Quack-v1 and Hyperquack-v1, if a vantage point responded incorrectly to control probes, it would be completely removed from the scan. Since Hyperquack-v2 is continuously running, we have made it so a vantage point that fails one of the intermittent ‘health checks’ that Hyperquack-v2 performs has the potential to come back after a user-defined period of time. This will allow for greater coverage in cases where a vantage point experiences momentary failure.

- **Ability for More Complex Scheduling** - This paradigm allows for far more complex scheduling of work than the previous system. In future, our goal is to produce a system where users that want a scan performed can submit the scan parameters to a scheduler server, which will then send that work to any number of worker servers, each running an instance of Hyperquack-v2. This paradigm will allow for multiple workloads to be scheduled simultaneously alongside any rapid response scans that crop up.

Below is a list of the other major changes we’ve made to Hyperquack-v2.

- **Combining Quack and Hyperquack** - Hyperquack-v2 combines the Quack and Hyperquack measurement methods by creating a standard interface for how internet protocols can be used for internet censorship measurement. With this interface, new protocols can be easily added to Hyperquack-v2.

- **Changes to Output Format** - In addition to the output from censorship trial, Hyperquack-v2 outputs the results of the previously mentioned ‘health checks’ from vantage points. This output is very similar to the trial output, with the change that if the ‘health check’ is passed, a template will be included. All responses from the vantage point will be compared to the template to detect interference. At the moment, the templates for the Echo and Discard protocols are pre-defined by the protocol, so only the HTTP and HTTPS protocols will have these dynamically-computed templates included.
1.3.2.1 Data Format

Trial Outputs

Trial outputs are produced when Hyperquack-v2 attempts to measure whether or not a vantage point observes the blocking of a given keyword by sending one or more probes to a vantage point.

Fields

- **vp** [String] The IP address of the vantage point used in this trial.
- **location** The location of the vantage point. This field has two subfields:
  - **country_name** [String] The name of the country the vantage point resides in, as provided by the MaxMind geolocation database.
  - **country_code** [String] The two letter code associated with the aforementioned country.
- **service** [String] The service of the vantage point we are using for this trial. This field is set to the name of the service. If the service is running on a non-standard port, a colon and the port number are appended (i.e., *discard* for discard on port 9, or *echo:8080* for echo on port 8080).
- **test_url** [String] The URL being tested by this trial.
- **response** An array representing the results of each probe sent to the vantage point. Each entry is a JSON object with six subfields:
  - **matches_template** [Boolean] Each trial performed by Hyperquack-v2 determines whether or not the probe was interfered with by comparing the response returned by the vantage point to an already known template. If the response does not match, that is potentially evidence of blocking. Set to true if the response given by the vantage point matches the known template, and false otherwise.
  - **response** [JSON Object] If the response given by the vantage point does not match the template, Hyperquack-v2 will add this field. Describes the response sent by the vantage point, including HTTP headers, the HTTP response code, the body of the response, and any TLS information.
  - **error** [String] If the probe fails with an error, that is potential evidence of blocking. If this occurs, this field will be included. Describes the encountered error. Note that this field can be used to filter out TCP handshake and setup errors.
  - **control_url** [String] During a trial, Hyperquack-v2 will sometimes send probes with non-sensitive URLs if all probes with sensitive URLs show evidence of being censored. If the probe described by this entry in the results array is a control probe, this field will be included. Contains the control URL used in the probe.
  - **start_time** [Timestamp] The time when the probe was sent.
  - **end_time** [Timestamp] The time when the response to the probe response arrived.
- **anomaly** [Boolean] Indicates whether the probes to the vantage point show enough evidence to conclude that the vantage point has observed some sort of anomaly, potentially indicative of blocking.
- **controls_failed** [Boolean] Set to true when all control probes sent to the vantage point fail to match the known template. This implies that the mismatching responses are due to an error in the vantage point or the network, not censorship. Rows with controls_failed set to true should not be considered for analysis.
- **stateful_block** [Boolean] Certain methods of censorship will block all communication from a given IP address for a length of time after that IP sends a request containing a censored keyword. We call this type of...
censorship ‘Stateful Blocking’. We detect this by sending a control probe immediately after our sensitive probes, waiting for some time, then sending another control probe. If the first control is blocked but the second isn’t, there is potentially stateful blocking. If this trial shows evidence of stateful blocking, this field is set to true.

- **tag** [String] If the work that produced this output was added to Hyperquack-v2 with a tag, this field will be included. Contains the work’s tag.

### Evaluation Outputs

Evaluation outputs are produced when the Hyperquack-v2 performs a health evaluation of a vantage point’s service. Services are evaluated by sending one or more probes containing control keyword to the vantage point.

### Fields

- **vp** [String] The IP address of the vantage point being evaluated.

- **service** [String] The service of the vantage point that is being evaluated. This field is set to the name of the service. If the service is running on a non-standard port, a colon and the port number are appended (i.e., `discard` for discard on port 9, or `echo:8080` for echo on port 8080).

- **response** An array representing the results of each probe sent to the vantage point. Each entry is a JSON object with five subfields:
  - **test_url** [String] The control URL used for this probe.
  - **response** [JSON Object] If the vantage point responds to the probe, this field is added. Describes the response sent by the vantage point, including the HTTP headers, the HTTP response code, and the body of the response.
  - **error** [String] If the probe fails with an error this field is included. Describes the encountered error.
  - **start_time** [Timestamp] The time when the probe was sent.
  - **end_time** [Timestamp] The time when the response to the probe finished arriving.

- **template** [JSONObject] If Hyperquack-v2 is able to generate a template from the probes, this field is included. Represents the expected response from the vantage point when sent a probe containing an uncensored keyword. If the service being tested is HTTP or HTTPS, this field is an HTTP response, including HTTP headers, the HTTP response code, and the body of the response. If the service is Echo or Discard, this field is omitted. This template is generated by the first probe during the health evaluation.

- **issue** [String] If there was an issue in generating the template for this service, this field will be included. Describes the issue encountered when generating the template or when comparing subsequent control probes to the template.

- **tag** [String] If the work that produced this output was added to Hyperquack-v2 with a tag, this field will be included. Contains the work’s tag.
1.3.3 Notes

While Hyperquack-v2 includes multiple trials intended to avoid random network errors, there is still a possibility that certain measurements are marked as anomalies incorrectly. To confirm censorship, it is critical that the raw responses are compared to known blockpage fingerprints. The blockpage fingerprints currently recorded by Censored Planet are available here. Moreover, network errors (such as TCP handshake and Setup errors) must be filtered out to avoid false inferences. Please use our analysis pipeline to process the data before using it.

Censored Planet detects network interference of websites using remote measurements to infrastructural vantage points within networks (e.g. institutions). Note that this raw data cannot determine the entity responsible for the blocking or the intent behind it. Please exercise caution when using the data, and reach out to us at censoredplanet@umich.edu if you have any questions.

1.4 Censored Planet Data Analysis

We provide two complementary ways to analyze Censored Planet data. The first is a fully operational data analysis pipeline (open-source) developed in collaboration with Jigsaw which requires a google cloud account and may be more expensive, depending on the amount of data. Additionally, we are also offering Beta access to an open-source dashboard built with the help of the Censored Planet data analysis pipeline, for easy exploration of data. The second is a set of analysis scripts that are more useful with analyzing a single scan, and understand how to work with the data.

1.4.1 Censored Planet Data Analysis Pipeline

The Censored Planet data analysis pipeline, developed in collaboration with Google’s Jigsaw, takes raw data from the Censored Planet Observatory and runs it through a pipeline to create bigquery tables for easier data analysis.

Because of the size of the data involved (many TB) this project requires a devoted Google Cloud project to run in. It is not recommended for end users to run the full pipeline because of cost considerations, but the code is made available for anyone who wants to understand how the data pipeline works. A small portion of the data can be used for development and testing purposes.
1.4.2 Censored Planet Dashboard

The Censored Planet Dashboard, built in collaboration with Jigsaw Inc. is an exploratory data dashboard that uses data analyzed using the Censored Planet data analysis pipeline, and contains visualizations that allow easy exploration of Censored Planet measurements. The dashboard is currently in beta stage, please reach out to censoredplanet-analysis@umich.edu if you are interested in using the Censored Planet dashboard for your research.
1.4.2.1 FAQ

Q. How do I access the dashboard?  
A. We are currently in the beta release phase for the dashboard, and we are only providing access for a limited set of users that can use the data and provide feedback. If you are interested in using Censored Planet data for your research, please contact us at censoredplanet-analysis@umich.edu and provide an overview of your research project and how access to the dashboard would be useful to you.

Q. What data is available on the dashboard?  
A. Currently, the dashboard contains data from Censored Planet’s Hyper-quack measurement technique, which measures network interference on the HTTP, HTTPS, Echo and Discard protocols. The measurements on the Echo and Discard protocols also aim to detect HTTP censorship. The two tabs on the top of the dashboard (see figure below) can be used to view the data related to HTTPS and HTTP tests. The DNS data from Satellite will be added to the dashboard soon.
Q. What domains does Censored Planet test and how often are the tests run? A. Every week, Censored Planet runs two identical measurements to each domains from the Citizen Lab Global Test List and a selection of popular domains from the Alexa Top Domains list on each protocol. The dashboard also contains some data from special measurements to other domains (including those in the Citizen Lab regional lists). Only domains are tested during the longitudinal scans, but full URLs can be tested in special scans.

Q. How are site categories calculated? A. Site categories are derived from Citizen Lab, and some websites are categorized manually. The site categorization may contain errors. Please submit a corrected category at https://forms.gle/78krbzEqr7vHYUrT6 if you notice any errors in site categorization.

Q. Can I filter by outcome? A. Yes, please use the outcome filter on the ‘How to use this dashboard’ pane (shown below) to filter certain outcomes. A brief description of outcomes is also provided.

1.4.2.2 Dashboard Walkthrough

We describe a demonstrative walkthrough of how to use the Censored Planet dashboard to characterize censorship in a country.

1. Select country and time range
   - Open the dashboard and select the country of interest.
   - The dashboard will show the data for the last 14 days. Select a different time range if you would like to analyze a specific event. Keep in mind that longer ranges make the dashboard take longer to load.

2. Clean up the networks

1.4. Censored Planet Data Analysis
• Identify the networks of interest. This requires knowledge of the local context. You should remove all networks that do not correspond to local ISPs so they do not interfere with the row order.
  
  – The Customers per AS table from APNIC may be useful if you are not familiar with the local ISPs.
  
  – Keep in mind that not all ISPs may be present because of a limitation in the collection methodology.

• Alternatively, you may select one Network at a time using the filter, but it helps to get a view across Networks first to see if there's consistency. Consistent blocking across Networks is evidence of a national firewall.

• Remove CDNs and private companies such as banks from the list of IP organizations. Banks tend to have a lot more censorship than ISPs and may use different methods.

3. Identify how each ISP blocks sites (see list of outcomes)

• In the outcomes per network chart, you can click on the “Optional metrics” icon and select “Unexpected count”. That will show what types of unexpected outcomes you get for each network.

• Keep in mind that not all mechanisms are measurable from outside the network, so the site may look unblocked when in fact it is.

• Near 100% of the unexpected probe count should be for a single outcome. If you see more than one unexpected outcome for a network, you may need to dig deeper. You can click on the down arrow to see the unexpected outcomes per subnetworks. If each subnetwork only has one unexpected outcome, you have characterized the censorship mechanisms for them.

• If you still see different outcomes in a subnetwork, it may be the case that different domains are blocked by different mechanisms. Reset the “Optional Metrics” to “Probe Count” and click on some of the domains and see if you get consistent results. If you get consistent results, you can drill back up to networks to see if they stay consistent within the network. That will give you a simpler view.

• If you still see inconsistent results, you should check the Outline Timeline. It may be the case that censorship for a domain changed during the selected time period.

• If the results are still inconsistent, you may need further investigation beyond the dashboard, and look at the raw data.

4. Identify the blocked websites and categories

• Click on the domains to confirm how and where they are blocked. For this it’s better to restore the “Optional metrics” to “Probe Count”. Take note of the site category they are in.

• You can click on the “+” button over the domains column to see their categories.

• As you identify blocked categories, you may exclude them from the Site Category filter to clean up the list. Pornography and gambling websites are often blocked and will monopolize the list.

• You can look for a specific domain using the Domain filter

5. Analyze both HTTP and HTTPS blocking. DNS and IP-based blocking are not available yet but will be added soon

6. It’s always helpful to try to confirm your observations with independent corroborating evidence from OONI, or by running your own probes (having access to the IPs and commands would help here). Make sure the other data sources report the same outcome you’ve identified in the Censored Planet data.
1.4.3 Censored Planet Data Analysis Tool

The Censored Planet Github repository provides some sample data analysis scripts that can be used to analyze specific Censored Planet scan data files. Currently, the analysis tool only supports analysis of v1 data. To install and run the analysis tool, please follow the instructions in the readme file. Please keep in mind that the analysis tool is very basic and limited, and cannot be used to do advanced analysis. The tool is more useful for understanding the steps needed to analyze Censored Planet data, and is useful for exploration. We plan to keep improving the analysis tools. Currently, the analysis tool offers three modes of csv output - blocked websites per country (aggregated), blocking per vantage point per country (aggregated), blocking type for website per country (aggregated). The output will show the number of measurements, number of anomalies, and number of confirmed cases of blocking. Anomalies may or may not be censorship, and will need more exploration. An example of how to analyze Censored Planet data to study which websites are blocked in Russia is shown below.

- Step 1 - Clone the repository

```
ram@worker-14: censoredplanet $ git clone git@github.com:censoredplanet/censoredplanet.git
Cloning into 'censoredplanet'...
remote: Enumerating objects: 347, done.
remote: Counting objects: 100% (347/347), done.
remote: Compressing objects: 100% (190/190), done.
remote: Total 347 (delta 173), reused 301 (delta 132), pack-reused 0
Receiving objects: 100% (347/347), 100.94 KiB | 3.60 MiB/s, done.
Resolving deltas: 100% (173/173), done.
```

- Step 2 - Build the binary

```
ram@worker-14: censoredplanet $ cd censoredplanet/analysis && make
mkdir -p bin
go install cmd/analyze-cp.go
go build -o bin/analyze-cp cmd/analyze-cp.go
```

- Step 3 - Download the data file from the Censored Planet website

```
ram@worker-14: analysis $ ls
bin cmd CP_Quack-echo-2021-04-22-01-01-04.tar.gz Makefile pkg
```

- Step 4 - Download the Maxmind database from their website. For v2 data, this is not necessary.

```
ram@worker-14: analysis $ ls
bin cmd CP_Quack-echo-2021-04-22-01-01-04.tar.gz Geolite2-City.mmbd Makefile pkg
```

- Step 5 - Run the analysis script

```
ram@worker-14: analysis $ ./bin/analyze-cp --input-file CP_Quack-echo-2021-04-22-01-01-04.tar.gz --output-file echo-2021-04-22.csv
```

- Step 6 - Extract the data for Russia

```
ram@worker-14: analysis $ grep "RU," echo-2021-04-22.csv > russia-echo-2021-04-22.csv
```

1.4. Censored Planet Data Analysis
• Step 7 - The processed data is ready. Either parse through the data in spreadsheets or import it into a visualization tool. A visualization of the above example using tableau is show below.
INDICES AND TABLES

• genindex
• modindex
• search