OONI on M-Lab

Description of the organization

OONI is the Open Observatory of Network Interference. *ooniprobe* is tool developed by members of the Tor Project.

Description of the research goals

The ooniprobe tool aims to collect high quality data using open methodologies and Free and Open Source Software (FL/OSS) to share information about surveillance, censorship, and discrimination as it occurs on networks. OONI on M-Lab plans to use active tests, in line with M-Lab's mandates, to achieve the goal of collecting open data that can inform researchers and the public.

Review of M-Lab requirements
(Reference doc here)

- **Free and Open Source code**
  - *Requirements:*
    Central repository containing server-side code and a client-side reference implementation (containing at least a reference for the application-layer protocol deployed). Should be released under a permissive license approved by the Free

Software Foundation (FSF), Open Source Initiative (OSI), or Creative Commons (CC).

- *Timeline:*
  Release to M-Lab as soon as possible to allow M-Lab's standard access and review, which needs to precede a steering committee approval. Release publicly on the launch of the tool on M-Lab, or no later than a year from launch or when a paper is published based on the collected data (whichever comes first).
- *Rationale:*
  The scientific process requires a bedrock of openness. On this we can agree.

- **Publicly available raw data that does not collect PII**
  - *Requirements:*
    Ensure that all OONI data can be released, raw, to the M-Lab repository for public use. This means:
    - The data collected and released poses no privacy issues, even if it technically complies with M-Lab requirements
    - The data cannot be correlated with other sources to break privacy.

    M-Lab's standard mandate for data collection is that we *only* collect IP address and active test data (i.e. a record of how the network reacted to a synthetic stream of data). For mundane performance testing, this covers it. For OONI data, we should explore:
    - More robust anonymization, including anonymous reporting tagged by non-IP identifier (i.e. absolutely non-human identifying).
    - Possibility of using a Tor hidden service on M-Lab servers to enable secure and anonymous reporting.
  - *Timeline:*
    A robust plan to accomplish this needs to be drawn up and vetted as a part of the steering review process. The data must be released either immediately upon the tools' initiating on M-Lab, or within a year or on publication of a paper based on the data, whichever comes first
  - *Rationale:*
    Self-explanatory
- **Tests run against M-Lab are actively-initiated by the client**
  - *Requirements:*
    Define which OONI tests can and cannot run against M-Lab (and identify clearly where we're not sure). This should include listing:
    - Tests that can work with M-Lab out of the box, now (without worry about IP address collection, etc.)
    - Tests that can work with M-Lab assuming it's possible to enable anonymous reporting (unique identifier replacing IP address).
    - Tests that cannot, categorically, run against M-Lab (e.g. varieties of passive monitoring).
    - Tests where we're not sure (e.g. requiring a small change to M-Lab)

    We should also spend some time to determine whether the current M-Lab structures need to be tightened to accommodate OONI (for example, per the

above to ensure truly anonymous reporting if desired, obviating IP address collection).

- *Timeline:*
  ASAP, as the implementation plan will need to anchor with the understandings developed here.
- *Rationale:*
  Our goal is 360 degree scientific verifiability -- we need to ensure that what is collected on the platform, and thus what any conclusions are based on, are completely open, allowing replication.

Description of the experimental methodology

Include data gathered by server, data gathered on client, things logged and who has access to logs.
For more detailed documentation see: https://ooni.torproject.org/docs/
For details on the backend component see: https://ooni.readthedocs.org/en/latest/architecture.html#oonib
(soon to be https://ooni.torproject.org/docs/architecture.html)

Notes: Tests are divided into two categories, those which detect *Content Blocking* and those which detect *Network Tampering*. *Network Tampering* related tests require a *Test Helper.*

Test documentation guide

It is ideal to read the documentation of tests that is stored on https://ooni.torproject.org/docs/.
The documentation for tests can be found here: https://ooni.torproject.org/docs/tests/.
Such pages contain a link to the *NetTest* source code (the *ooniprobe* client component) and, when required, the *Test Helper (*the *oonib* backend component).

Measurements

OONI tests that can run on M-Lab now

Common to all tests:

Data that must be collected on M-Lab and published.
The ISP of the client.
The region of the client (ASN).
Timestamps of start and end of test.
Two way traceroute.
What data would be nice to be collected on M-Lab and published?
The IP address of the client.
Full pcap from client.

Full pcap from server.

What is logged on the client?

Application state.

Who has access to client's logs?

Person running client only.

HTTP Invalid Request Line: https://ooni.torproject.org/docs/tests/http_requests.html

How does it work, briefly?

The goal of this test is to do some very basic and not very noisy fuzzing on the HTTP request line. We generate a series of requests that are not valid HTTP requests.

The remote backend runs a TCP echo server. If the response from the backend does not match with what we have sent then we say that tampering is occurring.

The idea behind this is that certain transparent HTTP proxies may not be properly parsing the HTTP request line.

Unless elsewhere stated 'Xx'*N refers to N*2 random upper or lowercase ascii letters or numbers ('XxXx' will be 4).

For more details see: https://ooni.torproject.org/docs/tests/http_invalid_request_line.html#test-random-invalid-method

Pending qsts for future work:

What data must be collected on the M-Lab server and published?

https://ooni.torproject.org/docs/tests/http_invalid_request_line.html#sample-report

What data does it gather from the client?

The synthetic HTTP Requests performed by the client and the requests that the server sees.

What data does it gather on the server?

NA

What is logged on the client?

NA

Who has access to client's logs?

NA

HTTP Header Field Manipulation: https://ooni.torproject.org/docs/tests/http_header_field_manipulation.html

How does it work, briefly?

It performes HTTP requests with request headers that vary capitalization towards a SimpleHTTPChannel test helper backend. If we detect that the headers the

backend received matches the ones we have sent then we have detected tampering.

Pending qsts for future work:

What data must be collected on the M-Lab server and published?
https://ooni.torproject.org/docs/tests/http_invalid_request_line.html#sample-report

What data does it gather from the client?
The synthetic HTTP Requests performed by the client and the requests that the server sees.

What data does it gather on the server?
NA

What is logged on the client?
NA

Who has access to client's logs?
NA

HTTP Host: https://ooni.torproject.org/docs/tests/http_host.html

How does it work, briefly?
This test is aimed at detecting the presence of a transparent HTTP proxy and enumerating the sites that are being censored by it.
It places inside of the Host header field the hostname of the site that is to be tested for censorship and then determines if the probe is behind a transparent HTTP proxy (because the response from the backend server does not match) and if the site is censored, by checking if the page that it got back matches the input block page.
Why do content blocking?
Q: Why should be do content blocking measurements with this test when we have other tests that also do this?
A: Why not? Although you are correct that technically the two tests are equivalent even though the IP layer differs in the two tests.
Note: We may in the future remove the Content Blocking aspect of the HTTP Host test.

Pending qsts for future work:

- What if we cache data (e.g., store the FB homepage)? Would that be a copyright probl?

What data must be collected on the M-Lab server and published?
HTTP requests being made and the responses.
https://ooni.torproject.org/docs/tests/http_host.html#sample-report

What data does it gather from the client?
The requests the client makes and the responses from the server or the transparent HTTP proxy.

What data does it gather on the server?
NA

What is logged on the client?
NA

Who has access to client's logs?
NA

Traceroute: https://ooni.torproject.org/docs/tests/traceroute.html

How does it work, briefly?
This test performs a multi port, multiprotocol traceroute test towards a backend. The goal of such is to determine biases in the paths based on destination port.
We perform a traceroute with destination port 22, 23, 80, 123, 443.
The test report includes the RAW IP packets sent and received. If the user has disabled to include the source IP in the report then we will remove the source IP for sent packets and the dst IP for sent packets.
The logged sent and received packets are only the ones that are generated and received in userspace via the scapy super socket that relies on libpcap and libdnet.

Notes: we currently do not perform the traceroute from the backend to the client probe. The backend is therefore not required to run any particular kind of software.

What data must be collected on M-Lab and published?
https://ooni.torproject.org/docs/tests/traceroute.html#sample-report

What data does it gather from the client?
Traceroutes.

What data does it gather on the server?
Traceroutes.

What is logged?
NA

Who has access to logs?
NA

Note: may need to mask ip address, first and last hop. Also the client IP address will be present in the IP header embedded in the ICMP Time Exceeded messages sent by each intervening router. This is a TCP/UDP/ICMP traceroute - it performs multi-protocol traces simultaneously on the client side.

DNS Tamper: https://ooni.torproject.org/docs/tests/dnstamper.html

How does it work, briefly?

This test performs A queries to a set of test resolvers and a known good control resolver. If the two results do not match it will perform a reverse DNS lookup on the first A record address of both sets and check if they both resolve to the same name.

What data must be collected on M-Lab and published?
Domain names requested and results from experiment and control resolvers. Possibly whether the query was seen by the name server if that's under our control.
https://ooni.torproject.org/docs/tests/dnstamper.html#sample-report

What data does it gather from the client?
Domain names requested.
Results from experiment resolver.
Results from control resolver.

What data does it gather on the server?
NA

What is logged?
NA

Who has access to logs?
NA

Tests for which we can store data, but don't need to run on a slice

Bridge Tor: https://trac.torproject.org/projects/tor/wiki/doc/OONI/Tests/BridgeT

How does it work, briefly?
A Tor client uses a geoIPdb to determine likelihood of connections to the Tor network being blocked, and then automatically iterates through a set of types of connections to Tor bridges, ranging from ICMP Echo (Ping) to a full Tor protocol connection.

What data must be collected on M-Lab and published?
Status per bridge per connection type. What kind of measurements were made and their result.
What data does it gather from the client?
Which Tor bridges were reachable/unreachable using which connection type. The application level view of what request was made (e.x. did an Ping, did a Tor connection). In the case of the Tor test we want to collect the info level log of Tor.
What data does it gather on the server?
None, there is no server component. However, it could be useful to run a “canary” Tor bridge on a server along with the triggering mechanism for a DPI probe, and then log connections by probes to canaries.
What is logged?
Which Tor bridges were reachable/unreachable using which connection type.
Who has access to logs?

HTTP Requests: https://ooni.torproject.org/docs/tests/http_requests.html

How does it work, briefly?
This test perform a HTTP GET request for the / resource over the test network and over Tor. It then compares the two responses to see if the response bodies of the two requests match and if the proportion between the expected body length (the one over Tor) and the one over the control network match.
If the proportion between the two body lengths is <= a certain tolerance factor (by default set to 0.8), then we say that they do not match.
The reason for doing so is that a lot of sites serve geolocalized content based on the location from which the request originated from.

What data must be collected on M-Lab and published?
The content of the sites being contacted, the request being made and the HTTP headers in the response. Eigenvector and value.
What data does it gather from the client?
The request being made and the response.
What data does it gather on the server?
None.
What is logged?
NA
Who has access to logs?
NA
Note: doesn’t need to run on M-Lab server, but could go to server and be stored in M-Lab repository. If we need something running on the server could implement.

Data collected and data collection pipeline

We will have two data collection “profiles”. One is more privacy aware and the other collects as much data as possible.

Privacy Aware Mode

In this mode we will only collect and publish ASN of the client and the geographical region (up to city level). We will also collect all the data that is available from an application level (on both the client and server side), this data must not contain any personal identifying information (this includes IPs).
The data collected on the Application level is basically the content of an OONI Report.

Privacy related settings may be configured via the *ooniprobe.conf* configuration file. See the privacy section of the sample configuration file: *https://gitweb.torproject.org/ooni-probe.git/blob/HEAD:/ooniprobe.conf.sample*

Full Collection Mode

In this mode of operation we will collect all of the above, but also a full pcap log from the server and client POV.

More details on data collection can be found here: https://trac.torproject.org/projects/tor/wiki/doc/OONI/DataCollection

Example of the data files

```
########################################
# OONI Probe Report for HTTP Host test
# Sun Nov 11 13:27:15 2012
########################################
---
{probe_asn: AS1267 WIND Telecomunicazioni S.p.A., probe_cc: IT, probe_ip:
151.0.0.1,
  software_name: ooniprobe, software_version: 0.0.7.1-alpha, start_time:
1352633236.0,
  test_name: HTTP Host, test_version: '0.2'}
...
---
input: torproject.org
report:
  request:
    body: null
    headers:
      Host: [torproject.org]
```

```
      User-Agent:
      - [Opera/9.20 (Windows NT 6.0; U; en), 'Opera 9.2, Windows Vista']
    method: GET
    url: http://127.0.0.1:1234
  response:

body: '{"request_method": "GET", "request_uri": "/", "request_body": "", "req
uest_headers":

{"Connection": "close", "Host": "torproject.org", "User-Agent": "(''Opera/
9.20
      (Windows NT 6.0; U; en)'', ''Opera 9.2, Windows Vista'')"}}'
    code: 200
    headers:
    - - ETag
      - ['"8e6968348dc5cc210fd4425c49a086e94353eebd"']
    - - Content-Type
      - [text/html; charset=UTF-8]
    - - Server
      - [cyclone/1.0-rc13]
    length: 219
    version: 219
  trans_http_proxy: false
  url: http://127.0.0.1:1234
test_name: test_send_host_header
test_started: 1352636836.020093
...
---
input: ooni.nu
report:
  request:
    body: null
    headers:
      Host: [ooni.nu]
      User-Agent:
      - [Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR
1.1.4322; .NET
          CLR 2.0.50727; .NET CLR 3.0.04506.30), 'Internet Explorer 7,
Windows XP']
    method: GET
    url: http://127.0.0.1:1234
  response:
```

```
body: '{"request_method": "GET", "request_uri": "/", "request_body": "", "req
uest_headers":
      {"Connection": "close", "Host": "ooni.nu", "User-Agent": "(''Mozilla/
4.0 (compatible;
      MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET
CLR 3.0.04506.30)'',
      ''Internet Explorer 7, Windows XP'')"}}'
    code: 200
    headers:
    - - ETag
      - ['"c82a5ec12b24d781f1f745ac7df224652085c7fb"']
    - - Content-Type
      - [text/html; charset=UTF-8]
    - - Server
      - [cyclone/1.0-rc13]
    length: 297
    version: 297
  trans_http_proxy: false
  url: http://127.0.0.1:1234
test_name: test_send_host_header
test_started: 1352636836.024071
...
```

Timeline for deploying the tool

Open questions

Running Tor bridge on M-Lab

Not used for “real” Tor node, but to test new builds.

PROS
- Stable up
CONS
- Don’t want M-Lab to become a Tor bridge

Collecting (and publishing) client IP address

- 2 modes to run OONI, based on user's consent.
    - Don't collect/publish client IP address and pcap.
    - Collect/publish client IP address and pcap

What if the servers get compromised?

- What do we expect to happen?
- Who may we want to talk to regarding SRE?
  Any possible Google SRE hours to help with sysadmin/security stuff?
  Two amazing people worth asking for help:
  Ben Kochie <ben@nerp.net> <- Google SRE guy
  Morgan <mmb@google.com> <- Google security incident response

Server whitelisting

The issue is that ISPs may be whitelisting certain IP addresses (the M-Lab ones) and behave differently if they see those.
This problem is very similar to the problem of Tor Bridges. Getting people that are running tests

- Not publish IP addresses for some time right after the initial deployment of every site.
    - How do we get these IP addresses to clients?
        - 1 per XX DNS resolution
        - hard-coded in clients
        - m-lab-ns returns 'private' fqdn
- Control servers for a subset of tests
- Inject honeypot server IP (well trafficked site) to results/DONAR/mlab-ns and watch the spike in throughput/latency/QPS.
- Provide SSL/TLS HTTPS server with valid cert (issued to some subdomain of an mlab domain) (i.e. redirection to a registered M-Lab service)

Deployment and user-consent

M-Lab server selection

https://docs.google.com/a/google.com/document/d/1eJhS75EZHDLmC6exggStr_b1euiR24_MVBJc1L6eH2c/edit#heading=h.ubhijxy606i

How to publish the data

M-Lab currently publishes data

- Raw format via Google Storage https://storage.cloud.google.com/?pli=1#m-lab
- Queryable format via BigQuery https://developers.google.com/bigquery/docs/dataset-mlab