Integrating Ooni with *mlab-ns*

This document lays out the components required for integrating *mlab-ns* with Ooni. It also describes who will be responsible for which parts.

Components

Backend Script

The backend script pulls the required information out of the *ooni-backend* installed to an M-Lab slice. It will be used by Nagios to obtain the information. Nagios will insert that information into *mlab-ns*.

Least Authority will partially complete this script. Our goal is to make the script obtain the information and feed it directly to our mock *mlab-ns* (see the section below). The script will have to be modified by the M-Lab team so that the information goes into Nagios instead.

Bouncer Script

The bouncer script runs periodically (cron job) on the slice hosting the Ooni bouncer. It pulls the aggregate of all information from the backend scripts, and provides that information to the bouncer so that it can be combined into a single configuration file.

Least Authority will partially complete this script. We will write it so that it works against our mock *mlab-ns*, freely assuming the required modifications to *mlab-ns* will be made. The M-Lab team may need to modify the script if *mlab-ns* is not modified as planned.

Mock *mlab-ns*

The mock *mlab-ns* will serve as our development / testing platform, since we cannot make modifications to the real *mlab-ns*. It will provide just two APIs:

- One API that accepts information from the backend scripts and stores it internally. This API will not be implemented in the real *mlab-ns*, but instead is a placeholder for the Nagios collection that we leave to M-Lab engineers.
- Another API that, when queried, returns a list of information about all Ooni backends. This API will be implemented in the real *mlab-ns*, so we will try to make it behave as similar to the real *mlab-ns* will as possible. Ideally, once the modifications are made to *mlab-ns*, it should just be a matter of changing the URL in the script.

Information

An unresolved question is: Which information needs to go in *mlab-ns*. We have several proposals for this, in order from "best engineering practice", to "dirty ooni-specific hack."

Proposal #1: Arbitrary Data

In this proposal, Nagios would accept an arbitrary string from the backend script, insert it into *mlab-ns*, and then *mlab-ns* will be modified so that it's possible to query for a list of all Ooni backends and their associated arbitrary data. This is nice, because the modifications to *mlab-ns* are simple (it doesn't need to understand the arbitrary data), and it will be useful for all future experiments running on M-Lab, not just Ooni.

Proposal #2: Complete Information, but *mlab-ns* parses it

In this proposal, Nagios would parse out the separate fields of information from the backend script, and *mlab-ns* would be made to understand them all. The information required, with their associated data types, is, for each backend:

- A *.onion* address (string).
- A list of supported test helpers (list of string).
- A list of test helpers (each a name string, IP address, and port number).

This is not ideal because *mlab-ns* would have Ooni-specific modifications, that aren't useful for future experiments.

Proposal #3: Limited Data for First Deployment

Another option is a "hack" which would allow Ooni to be deployed with just one test helper, and just one collector policy. Nagios would have to insert into *mlab-ns* the following information:

- One *.onion* address. The list of supported test helpers will be *assumed* to be just the HTTP return headers test.
- One IP address. It will be *assumed* that on port 80 an HTTP return headers test is running.

This would be adequate for the first deployment, but it is obviously not sustainable and does not allow Ooni to grow. We believe that implementing this and Proposal #1 would incur a similar cost, so we prefer Proposal #1.