OONI : Open Observatory of Network Interference

Abstract

OONI, the Open Observatory of Network Interference, is a global observation network which aims is to collect high quality data using open methodologies, using Free and Open Source Software (FL/OSS) to share observations and data about the various types, methods, and amounts of network tampering in the world.
With the belief that unfettered access to information is a intrinsic human right, OONI seeks to observe levels of surveillance, censorship, and network discrimination in order for people worldwide to have a clearer understanding of the ways in which their access to information is filtered.
The end goal of OONI is to collect data which can show an accurate topology of network interference and censorship. Through this topology, it will be possible to see what the internet looks like from nearly any location, including what sites are censored, or have been tampered with, and by whom. We're calling it filternet.

1 Introduction

Previous work in the realm of network filtering detection was mainly focused on either more general network measurements with the goal of studying network quality[1] or in identifying the presence of traffic discrimination devices. Tools and techniques specifically dedicated to detecting censorship are either not open source software or the results from the experiments are not published in todo.
The major point that distinguish OONI from existing filtering detection technology is that it's a framework that allows researchers to build on top of their own censorship detection tests. The methodologies used and the data collected are all open, allowing journalists and researchers alike to write network filtering related reports that are based on independently verifiable evidence.
OONI aims at detecting the presence of a passive network interception device that do traffic discrimination and at understanding what content is being targetted for discrimination. We wish to understand what addresses are being blocked, the keywords being filtered and the kinds of protocols being biased.
This paper aims at being an introduction to the concepts and design choices we made in developing the OONI framework, though it still leaves a lot open questions that will be the topic of future work.

2 Goals

The goal of OONI is to detect network related tampering, this means in general network interception of which internet surveillance is a subset. OONI-probe wishes to detect the presence of network traffic manipulation from an edge network perspective. It also wishes to obtain, when that is the case, the type of content that access is being restricted to (what websites are blocked, what keywords are being filtered).

When possible we will also attempt to profile the interception device in an attempt to identify the vendor and product being deployed.

2.1 Open Data

It is very important that all of the data collected by OONI is released to public under an Open license. This allows anybody to freely use and republish this information as they wish without any restrictions. The publishing of scientific data is in line with the Open science data movement. We strongly believe that the rate of discovery and scientific progress is accelerated by better access to data.[2]

2.2 a FLOSS tool

Making to tool available as Free Libre Open Source Software allows researchers to fully grasp it's inner workings and it allows the building of a strong skillful community around OONI. People interested in expanding it will be able to do so freely and the lifetime of the project is grossly extended. If we stop working on it hopefully somebody else will pick the project up and maintain it.

2.3 Open Methodologies

Only by having openly known and peer reviewed methodologies are we able to produce data that is of scientific value. We believe in the importance of reproducable research[3], this means that any researcher should be able to obtain the same results indipendently. This can only be achieved by detailing the methods we use in our experiments in plain english as well as in code. We plan to set the standard as to which should be the best practices when developing networking filtering detection tests.

2.4 Extensible tool

The tool should be extensible so that any researcher interested in trying out their ideas can bootstrap the process of writing the test by using the OONI-probe framework. The framework is written in python and provides base classes and methods that can be extended to suite the needs of the researcher. It is designed around the concepts of non-blocking network I/O allowing multiple tests to run efficiently in parallel. If running tests in parallel may pertubate the network to a point that the test can result inexact (this may be the case in latency based measurements) it is possible to trade off efficiency for accuracy.

2.5 Community

In order for this field to progress it is necessary to build a community around network filtering detection. We hope that by making this software and data available to the public a variety of people interested in the topic will gather around OONI. People that could be interested are researchers developing network filtering detection tools, social scientists that wish to have high quality data on censorship around the world, data visualization specialists that wish to make the general public access this data in an easy to grasp way.

2.6 Awareness

Another important goal of OONI is that of raising awareness in the general public on the topic of censorship. Data driven journalism is a form of journalism that is based on processing large data sets for the purpose of creating a story. With OONI we plan to supply this new generation of journalists with high quality data that they can use to build their stories on. This allows technically proficient people to focus on the technical aspects while still providing indirectly information that is graspable and undestandable by the general public.

3 Threat Model

Our adversary is capable of doing country wide network surveillance and manipualtion of network traffic.

The goals of our adversary are:

- Restrict access to certain content, while not degrading overall quality of the network
- Monitor the network in a way that they are able to identify misuse of it in real time

More specifc to the running of network filtering detection tests:

- Detect actors performing censorship detection tests
- Fool people running such tests into believing that the network is unrestricted

Note that while 2) =¿ 1) it is not true that 1) =¿ 2) as the identification of such actors does not necessarily have to happen in real time.

While our intention is to minimize the risk of users running OONI probe to be identified, this comes with a tradeoff in accuracy. It is therefore necessary in certain tests to trade-off fingerprintability in favour of tests accuracy.

This is why we divide tests based on what risk the user running it can face, allowing the user to freely choose what threat model they wish to adere to.

4 Methodology

We intend to apply the *scientific method* to the realm of network filtering detection. Our goal is to make our results as objective as possible. To achieve this all the experiments that are run shall be properly documented and the data collected made available to the public. This means that the same observations can be reproduced indipendently in line with the *full disclosure* practice.

We base our tests on the concepts of the experiment and the control. The experiment is what is being run on the *test network* and the data is stored as the experiment result. The control is what is what the expected result should be and it is compared with the experiment result. If experiment and control mismatch this is an indication of some unusual network activity.

Keep in mind that mismatch between experiment and control is not always a clear signal of network manipulation, but they are a clear indication that something wrong may be going on. We will always favour false positives rather than false negatives. This means that it's better to have more events that indicate the presence of censorship rather than fewer. This is because the false positives can then be investigated further and the researcher is able to understand if censorship is in fact happening.

There are instances in which the experiment-control methodology cannot be applied and in these cases the researcher is still advised to focus on being in favor of a higher false negative rating.

Every test should include a high level description of how the tests will work as well as a technical in depth description. In describing the methodology we will focus on what the result actually means and how accurate the result should be considered to be. The reason for this is to give also a non technical audience the ability to grasp the actual significance of such a test and how accurate they should consider the result.

The methods will also be classified by level of risk, in the sense that the risk the person running the test on their network should be quantified. This allows us to be completely transparent with people that wish to support the project by running tests on their network. By making it clear what risks they may incur into by running a test they are able to make a consented decision.

5 Architecture

OONI revolves around these major concepts: Assets, Tests, Reports, Work Units, Nodes and Backends.

The client is responsible for running OONI-probe and this will lead to taking measurements on the selected Node. The Node in question can either be local (the measurement is being done on the same network as the client is running) or it may be remote. Backends run the server side part of OONI-probe and they are used to make baseline measurements that require bidirectional communication.

The tests that are run by OONI can be divided into two macro categories: Network tampering detection and Content censorship detection.

For network tampering detection tests there is no need to supply a list of assets or targets to be tested for blocking, in content censorship detection this is required.

In the second case the process of running an OONI scan requires an extra pre processing stage. The aim of this phase is to collect a set of hostnames, URLs and/or keywords that are likely to be censored on the target network.

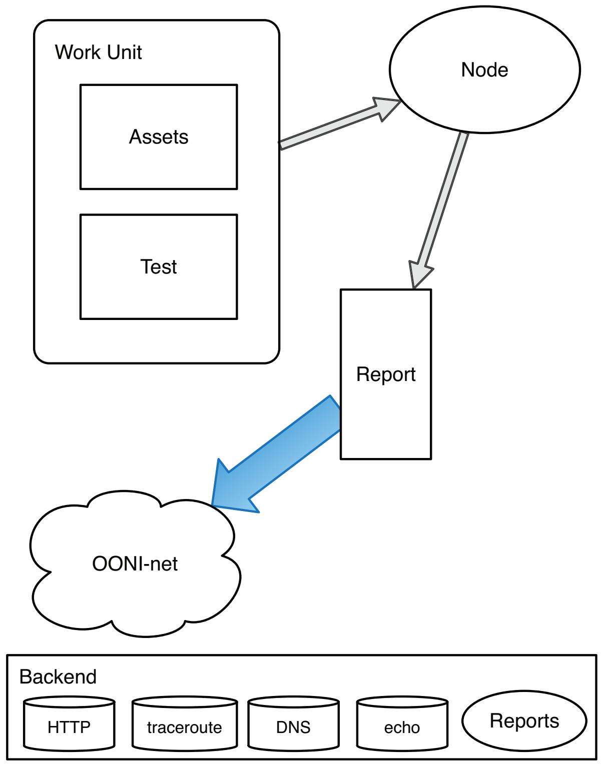

Figure 1: OONI Architecture design

5.1 Pre processing

This is a phase in our methodology that aims at building Assets to be run on the tested network. For example if I am interested in running a censorship detection test in an arab language, that is known to censor politically oriented sites my list of addresses should contain sites of this kind.

Assets shall be classified by language and category. To obtain such a list we will use a mix between agregating open data on web sites (Alexa Top 1 Million), crawling the web for certain types of sites and collecting feedback from users on what should be tested.

5.2 Assets

Assets are the inputs used inside Tests to detect censorship events. These can be URL lists, keywords, ip addresses, packets or any kind of set of data. In the python specific implementation this is represented as a python iterable object. This means that the Testing framework will be able to iterate through every element in the Asset.

5.3 Tests

This is the core of OONI. These are the actual tests that will be run using as input (if an input is required) the Assets. Tests can be summarized as an experiment and a control. The control represents the expected result and the experiment is the network operation being performed on the live network. If the experiment does not match up with the control then a censorship event had occured.

OONI probe provides some useful functionality to the application developer that may be useful when developing censorship detection tests. For example it is possible to make a request over the Tor network easily or use a fast and flexible non-blocking HTTP client implementation.

5.4 Unit of Work

Unit's of work represent a number of Asset and a Test associated to them. It is useful to split tasks to be run by OONI into unit's of work to allow resuming and parallilization accross multiple hosts. It may be of interest for example to split the task of going through a 1 million entry URL list to 10 machines in a certain country. This allows for a faster detection of censorship events.

5.5 Reports

This is the data that is collected from the test. OONI probe provides a flexible means of storing results and uploading this data to a remote server or a flat file.

The Test developer should include in the report as much data as possible and can contain raw packet dumps as well as structured synthetic results.

In future on top of ooni-probe Reports it will be possible to develop flexible post-processing tools to allow data-visualization guru's to properly visualize and contextualize the resulting data.

5.6 Nodes

Nodes are network enabled entities capable of running OONI-probe tests. There are currently two kinds of Nodes: OONI Nodes and Network Nodes.

5.6.1 OONI Nodes

OONI Nodes are ones capable of accepting OONI-probe commands and executing tests on their own. They run OONI-probe software and they can be asked to run a set of tests and just notify us back once they have completed. OONI Nodes take as input what a Unit of Work and upon completion will report to the OONI-net the result of it's execution.

5.6.2 Network Nodes

Network Nodes are any kind of device that allows to route traffic through them. They should be used as last resort impromptu systems for testing censorship in certain countries. Nonetheless they can be very valuable. It is quite easy to obtain big lists of open SOCKS proxies in almost every country and when that is not the case it's trivial to build one up by port scanning on port 1080. The issue with using network nodes is that it can be difficult to understand if the test is failing because the proxy went bad or because a censorship event happened. For this reason it is advisable to keep sending in the background at a 5 hrz interval packets to a backend echo server as suggested Kreibich et al in.[1]

5.7 OONI-net

OONI-net is the backend network responsible for helping OONI-probe run it's tests. On these machines we have common services running such as DNS, HTTP, TCP and UDP echo, bidirectional traceroute and also the reporting infrastructure. The backend infrastructure should be distributed and extensible. On startup the client will pick a backend node from OONI-net and use this node for running tests that require a backend.

6 Censorship detection tests

We will give a high level description of the kind of tests that OONI-probe will be shipping with though because of length constraints will not go into great detail.

The Test are divided into two main categories: Network tampering detection, Content censorship detection.

6.1 Network tampering detection

These are tests whose aim is to detect the presence of deep packet inspection devices (DPI). Their aim is not that of detecting what is being censored, but rather if and how.

Tests that fit into this category are:

Two way traceroute If there is a difference between an inbound traceroute and an outbound traceroute for certain source and destination ports this may be an indication of traffic being routed to interception devices.

Header field manipulation By varying the capitalization and adding certain headers to layer 7 protocols it is possible to detect on the receing end if the traffic has been tampered with.

6.2 Content censorship detection

The goal of these tests is to detect what content access is being restricted to. These tests usually involve going through a list of keywords or hostnames to identify which subset of these are being blocked.

Examples of tests that are in this category are:

HTTP Host This involves changing the Host header field of an HTTP request to that of the site one wishes to check for censorship.

DNS lookup This involves doing a DNS lookup for the in question hostname. If the lookup result does not match the expected result the site is marked as being censored.

Keyword filtering This involves sending an receiving data that contains certain keywords and matching for censorship. It is possible to use bisection method to understand what subset of keywords are triggering the filter.

HTTP scan This involves doing a full connection to the in question site. If the content does not match the expected result then a censored flag is raised.

Traceroute This involves doing TCP, UDP, ICMP traceroute for certain destination addresses if there are discrepancies in the paths with locations in the vicinities then a censorship flag is raised.

RST packet detection This involves attempting to connect to a certain destination and checking if the client gets back a RST packet.

7 Data Collection

All the data collected by OONI-probe must be open and accessible. We will do the effort to not redact any data that is contained in the logs, unless it's content could potentially lead to identification of users and lead to privacy leaks.

The data format for the reports is YAMLOONI. Every report must contain as a bare minimum the following information:

- Timestamp of start and end of test
- ASN from which the test originated
- The address from where the test was run (optional)
- A bidirectional traceroute from and to another OONI-probe node (optional)

The bidirectional traceroute should be done with source and destination ports set to 0, 21, 80, 123, 443, UDP, TCP and ICMP. If A is the host from which the test is being performed the host will pick a random OONI-probe node X and perform a traceroute to X. X will be signaled that they need to traceroute to A and they will run the same traceroute and send the result to A. This is useful to understand the topology of the network from which the test is being run from. This traceroute can potentially leak information about the fact that the user is running OONI-probe software and should be run once all of the tests that needed to be run have completed.

The timestamp format we use is that specified in RFC3339. All times are expressed in UTC.

This is an example of a YAMLOONI OONI-probe report:

```
# OONI Probe Report for Test httphost
# 18th of April 2012 18:00:00
---
test_name: httphost
asn: ASN-59395
addr: 198.51.100.1
start_time: 2012-04-18T18:00:00.00Z
---
start_time: 2012-04-18T18:00:00.00Z
end_time: 2012-04-18T18:00:02.12Z
result: {'thetestresult': 'data'}
---
start_time: 2012-04-18T18:00:00.00Z
end_time: 2012-04-18T18:00:03.12Z
result: {'thetestresult': 'data'}
# Test ended in 200s
---
end_time: 2012-04-18T18:00:00.00Z
traceroute:
- dst: 80
  src: 80
  tcp:
  - [10.0.2.1, 794.792, 6.323, 1.18]
  - [198.51.100.2, 25.092, 11.716, 44.371]
  - [203.0.113.5, 59.241, 12.302, 14.776]
  - [203.0.113.8, 59.241, 12.302, 14.776]
  timestamp: '2012-04-18T19:11:14'
  udp:
  - [10.0.2.1, 794.792, 6.323, 1.18]
  - [198.51.100.2, 25.092, 11.716, 44.371]
  - [203.0.113.5, 59.241, 12.302, 14.776]
  - [203.0.113.8, 59.241, 12.302, 14.776]
```

We decided to choose YAML as a data format since we believe it is the best compromise between human readable and machine parsable. YAML supports binary data allowing us to store also packet dumps inside of reports.

8 Community

It is very important to build around OONI an active community of both developers, social scientists, journalists and people interested in working on the topic of internet censorship and surveillance. These people, with a varied set of skills, will all contribute to the growth of the project.

Developers will be interested in OONI, because they will be able to write their own tests with minimum effort. People in the social science field will be able to use OONI-probe data to base their researches on and explore the context of countries where censorship in deployed. Journalists will be able to cite a trustworthy resource when they wish to do data journalism. People interested in sensibilizing the wider audience on the mater of censorship will be able to draw upon the OONI data to make their point.

9 Deployment

OONI-probe will be deployed mainly in two ways. Users will be invited to run an OONI Node or to use OONI-probe to do tests of network traffic manipulation in their country. They will be able to do so by downloading an application that they will execute on their machine. OONI-probe is written in python and will be cross built to support Windows, OSX and Linux alike.

Another means of deployment will be integrated inside of the Tor Router. This will allow people to help the project by just configuring their router to run the OONI-probe tests they feel confortable with running.

10 Limits and future work

We did not deal with a lot of issues inside of this paper because of size constraints. It is for example not clear how much data from running tests we should be collecting and if we should be properly anonymizing it.

We have not found a good solution to detecting the object of censorship when dealing with very large pools of addresses and searching for obscure sites amongst these (needle in a hackstack problem).

Our threat model is not very clear and we still are unsure if there will be active effort by ISP's in detecting network censorship detectors and if it is indeed worthwhile to focus on making it harder for them to detect the presence of an OONI-probe. By making it harder for the censorship to detect the detection attempt we will probably end up with slightly less reliable results, though this is still not exactly fully clear.

We also did not deal with any specific tests that we wish to run with OONI-probe. More details on the actual censorship detection tests will be given in future papers.

11 Conclusion

References

[1] Kreibich, C., Weaver, N., Nechaev, B., and Paxson, V. Netalyzr: illuminating the edge network. In *Internet Measurement Conference* (2010), M. Allman, Ed., ACM, pp. 246–259.

[2] Norris, R. P. How to make the dream come true: the astronomers' data manifesto. *Data Science Journal 6* (2007), S116–S124.

[3] on Data, Y. L. S. R., and Sharing, C. Reproducible research. *Computing in Science and Engineering 12* (2010), 8–13.

Notes