OONI : Open Observatory of Network Interference

Arturo Filastò
The Tor Project
art@torproject.org

Jacob Appelbaum
The Tor Project & The University of Washington
jacob@torproject.org

Abstract

OONI, the Open Observatory of Network Interference, is a global observation network which aims to collect high quality data using open methodologies, using Free and Open Source Software (FL/OSS) to share observations and open data about the various types, methods, and amounts of network tampering in the world.
Furthermore, OONI is a human rights observation project – observation is a fundamental requirement for the advancement of knowledge and OONI aims to ensure that the tools to make such observations are freely available to all. With the belief that unfettered access to information is an 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 and speech is monitored, censored or otherwise filtered.
The end goal of OONI is to collect data which will show an accurate topology of network surveillance, interference and outright censorship. Through this data, it will be possible to draw conclusions about how the internet functions from any location where an OONI probe is present. This data includes which websites are censored, or which services have been tampered with, and by whom. The data also includes information about the observer and will attempt to classify the results. We use the term filternet to describe network connections that are under measurable surveillance, tampering, or subject to censorship.

1 Introduction

Previous work [1] in the realm of censorship, network filtering, or network censorship detection has largely focused on general anecdotes [2], network measurements with the goal of studying network quality [3], in identifying the presence of traffic discrimination devices [4] or bypassing such censorship without a full understanding [5, 6, 7] of the system itself. Legal [8] issues have often caused hurdles [9, 10, 11, 6] for researchers. Tools and techniques specifically dedicated to detecting censorship are generally not FL/OSS software [12, 3], the technical methodology is usually secret [13], or the resulting experiment data is not published in totality, if at all [14].
The major point that distinguishes OONI from existing filtering detection technology is that it is a framework that allows researchers to expand upon it with their own surveillance and 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 to detect the presence of otherwise seemingly passive network interception devices which perform traffic discrimination and also to understand what content is being targeted for discrimination. We wish to understand what addresses are being blocked, the keywords which are filtered, and the kinds of protocols that are impacted.
This paper aims at being an introduction to the concepts and design choices we made in developing the OONI framework, although it still leaves a lot of open questions which will be the topic of future work. OONI is currently in heavy development and is available as Free Software [15] from The Tor Project.

2 Goals

OONI is designed to detect network related tampering and to assist in network topology discovery. OONI has a strong focus on general network interception, of which internet surveillance is a subset, and of which internet censorship is a further subset. OONI-probe wishes to detect the presence of network traffic manipulation from the perspective of a given edge network probe. The

OONI-probe software generally attempts to obtain the type and kind of content where access is restricted. It gathers local network information and is able to make high speed, bulk queries to the greater internet for various popular protocols, such as HTTP, HTTPS and DNS.

When possible, we will also attempt to profile any detected interception devices. We attempt to identify the vendor and product used in a given surveillance and censorship system by first locating the device and then by using various probing techniques. Unsurprisingly, many interception devices in the wild advertise their make and model information.

2.1 Open Data

It is very important that all of the data collected by OONI is released to the public under an Open license such as the Creative Commons by Attribution [16] license. This allows anybody to freely use and republish this information as they wish without restriction. The publishing of scientific data is in line with the Open Science data movement – the rate of discovery and scientific progress is accelerated by better access to data. [17]. While other projects have attempted to share data, we assert that high level summaries are simply not enough. We wish to ensure that open access to data submitted is available to all and that techniques for analysis will be equally available. This will allow for factual assertions based on the data collected rather than on organizational reputation alone. While we think that this model may have risks, no project has resolved such risks by refusing to publish their datasets. At best, hiding data may stop casual attackers, and, at worst, it creates a very valuable target for attack [18].

2.2 A FL/OSS tool

Making the tool available as Free Libre Open Source Software allows researchers to fully grasp the inner workings of the software, and it allows the building of a strong skillful community around OONI. People interested in expanding it will be able to do so freely, which will help sustain it as a basis for future work. A FL/OSS tool has the most likely chance of being studied and improved by the research community.

2.3 Open Methodologies

By having openly known and peer reviewed methodologies, we able to produce data that is of scientific value. We believe in the importance of reproducible research [19], meaning that any researcher should be able to obtain the same results independently. 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 standards for best practices in developing network surveillance and filtering detection tests.

Focusing on the creation and utilization of tests based on open methods will allow us to build a taxonomy of surveillance and censorship. This will expand the ability of the larger community to independently implement tests, perform experiments, and reference specifics.

2.4 Towards an Extensible Framework

OONI should be considered the basis of an extensible framework. Any researcher interested in trying out their ideas may bootstrap the process of writing tests by using the OONI-probe framework. This will ease data collection and correlation as well as providing a common functionality that each researcher would otherwise need to implement. The OONI framework is written in Python and provides base classes and methods which can be extended to suit the needs of researchers and users. It uses non-blocking network I/O, allowing multiple tests to run efficiently in parallel, and, if desired, blocking network I/O is a configurable option.

2.5 Enhancing Awareness

Another important goal of OONI is raising awareness in the general public on the topic of surveillance and censorship. Furthermore, legal analysis of specific realities [20, 21, 22, 23] will allow society at large to have a more open discussion. With OONI, we plan to supply this new generation of journalists and legal scholars with high quality data that they can use to build their stories upon. This allows technically proficient developers to focus on the technical implementation and low level details.

3 Threat Model

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

The goals of our adversary are:

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

Our adversary may wish to detect surveillance and censorship probing by detecting testers and they may even desire to tamper with the tester's results. The identification of testers does not necessarily have to happen in real time. While our intention is to minimize the risk of users running OONI-probe being identified, this may come with trade-offs in accuracy. It is therefore necessary, in certain tests, to favor fingerprint-ability over accuracy. As an example, rTurtle [13], the software produced by the ONI, connects to a centralized server [24]

with easily fingerprinted DNS and HTTPS traffic. We believe such simplistic fingerprinting must be avoided.

4 Methodology

We intend to apply the *scientific method* to the realm of network surveillance and filtering detection. In order to ensure reproducibility, all experiments conducted shall be properly documented and all data collected made available to the public in a timely manner. The same observations should be possible to reproduce independently, in line with standard *full disclosure* practice.

We base our tests on the concepts of experiment and control groups. The experiment is defined as the portion being run on the *test network*, and the resultant data is stored as the experiment result. The control is defined as what the expected result should be on an unbiased, uncensored, and otherwise untampered network, and this control data is compared with the experiment result. If experiment and control mismatch, then this is an indication of some unusual network activity. The control data may be dynamic or static – for example, some DNS records are predictable while many webpages are geographically diverse.

Mismatch between experiment and control data is not always a clear signal of network manipulation, but in many protocols, it is a clear indication that some kind of tampering has taken place. We will always favor false positives rather than false negatives. This means that it is better to have more events that indicate the possible 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, occurring. This may take the form of large scale data analysis across all sample data, or, perhaps, only against a subset of the data. Collection and analysis should be considered as separate phases even while we do some kinds of analysis during data collection.

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 an in-depth technical description. In describing the methodology, we will focus on result significance and accuracy, in order to enable non-technical audiences in grasping the actual meaning and how accurate they should consider the result.

The methods will also be classified by a quantifiable level of risk. In other words, any person running the test should be able to comprehend the visibility and type of traffic which will be generated, what information will be collected, stored, or sent, and how that information will be stored or sent. Therefore, given the context of their own political, economic, and legal circumstances, a person should be able to reasonably calculate a person risk assessment. We plan to do this by presenting a full text description of the methods and the data, allowing us to be completely transparent with people who wish to support the project by running tests on their networks. By making it clear what risks they may incur by running a test, users will be able to make informed consensual choices. We also intend to educate users about possible testing scenarios, and to ensure that they understand any potential differences in conducting tests under personally identifying circumstances, such as the difference between running a test on an open wireless network versus their cell phone with a SIM card registered to their passport. We believe that previous efforts have not produced such educational material and that users have been left in the dark.

5 Architecture

OONI is split into two main Modules: OONIB and OONIProbe.

OONIB is the backend component of OONI. It is responsible for exposing test helpers and for collecting reports from probes. Anyone may run an instance of either, and we believe this will ensure that the collection of probe data is resistant to simplistic denial of service. We plan to provide a public set of collection services over HTTPS and as Tor Hidden Services [25].

TestHelpers are network services that are exposed to OONIProbe clients. These could be, for example, an HTTP server, a DNS server, or a Traceroute server. When using the TestHelpers, the OONIB will also keep track of the testing session, logging all traffic related to the client's testing session and including it in the report. Once the client sends their side of the report, OONIB will match up the client submitted data with the data it collected locally.

OONI-probe is the measurement tool that will be run on edge networks; here resides the core of the test logic.

The Preprocessor takes files and inputs and makes them into Assets. Assets are the inputs to the Test.

The Test module takes care of writing packets to the wire based on the given inputs. It is designed to be extensible, and it exposes an interface allowing third party developers to write tests for it.

The PacketCapture module is responsible for collecting the packet dumps of a test. This can be achieved either by having the test writer specify a rule for collecting packets relevant to the test (BPF filters) or by creating a tun/tap device through which all the test related data is routed through.

The ControlChannel is used when running tests with the assistance of OONIB TestHelpers. As such, the

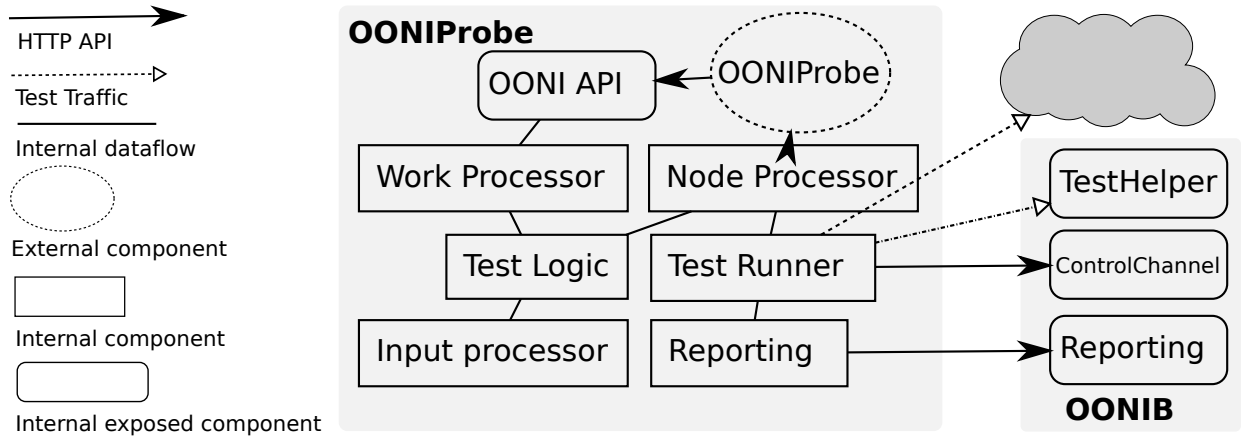

Figure 1: OONI Architecture design

ControlChannel is useful for out-of-band communication with the backend. During a testing session, the client may be interested in notifying the OONIB of the data they have sent. OONIB will verify that the data sent matches those received and report the result back to the client.

The WorkProcessor is responsible for accepting inputs from a remote OONIProbe client and for running tests obtained through it. If enabled, it exposes an HTTP API to receive remote test inputs.

The UtilityLib provides utility functions which are useful for test writing, for example, networking libraries.

Reporting may occur locally or remotely. In the case of remote reporting, the OONIProbe client will obtain from an OONIB a Test submission token. This token will allow them to communicate to the backend the progress of the Test. The reporting system is designed to accept data from third party testing systems. OONI does not deal with making measurements using web technologies (browser based solutions), however, it is possible to develop such systems and use OONIB's reporting system for submitting the data.

The TestState is used internally to keep track of test progress and state. This is useful for handling errors during testing and for providing resume support.

5.0.1 Nodes

The purpose of the Node module is to allow tests to be run on remote machines. A node can also be another OONIProbe that accepts incoming test requests. If that is the case, then the only information which needs to be sent over the network are the inputs to the test – all test logic will be present serverside.

Nodes which are not aware of OONI are called Network Nodes. Such Network Nodes are any kind of device that allows traffic to exit from them, and should be used as last-resort, impromptu systems for testing censorship in certain countries.

OONI plans to develop an extension to the SOCKS protocol that will allow for the creation of RAW sockets. An OONIProxy will then be Network Nodes which are not necessarily OONI-aware, i.e. they do not run any OONI software components, but are able to understand this extension to the SOCKS protocol.

5.1 Detection Procedure

1. The inputs to the test are obtained either from a remote network call or by passing the user-provided inputs to the Input Processor. This generates an OONI Asset. These items can be URLs, keywords, IP addresses and will be given as input to the test.
2. The inputs are passed to the test logic.
3. If the user has specified that the test should run on a remote Node, the specified portion of the inputs is serialized and sent to the remote OONI Node. If the specified Node is remote, a check, based on the test procedure, is made to ensure that the Node has the requisite capabilities for running the test, for example, a normal SOCKS server would not allow for raw socket access.
4. If the test requires communication with an OONIB, a new test session token is created.
5. The test is run on the network.
6. Every N results, a report is sent back to the OONIB.
7. Once the test has finished, the final report is sent to the reporting backend, and a report receipt is delivered to the client.

5.2 Test categorization

The tests which are run by OONI can be divided into two generalized categories: Traffic Manipulation and Content Blocking.

For Traffic Manipulation tests, there is no need to supply a list of assets or targets to be tested for blocking,

whereas in the case of Content Blocking tests, such inputs are required.

When running a Content Blocking test, the inputs go through a preprocessing phase to aggregate a set of any hostnames, URLs, and/or keywords which are being tested for censorship on the target network into an Asset.

5.3 Preprocessing

At this phase in our methodology, Assets, which will be passed to the Test, are constructed. For example, if we are interested in running a censorship detection test for a given country that is known to censor politically oriented sites, our list of addresses should contain sites in that category.

Assets should be classified by language and category. To obtain such lists, we will use a mix between aggregating open data, obtained by crawling for categorically relevant web sites, and collecting user's suggestions for what should be tested.

5.4 Test Specification

We believe in the importance of detailing tests in plain English as well as in code. The test should be specified before being implemented. We propose the following template for detailing an OONIProbe test:

What it detects Explain what the test aims at detecting.

Test Categorization If it is a Traffic Manipulation test or a Content Blocking test.

Inputs If the test requires any inputs and what those inputs should be.

Experiment The operations which will occur on the test network.

Control The operations which should verify if censorship has occurred.

Output The data which will be output from the test.

We will not go into the details of every test, but will provide a brief overview of the kinds of tests we plan to specify, implement, and run. The reader interested in more details on test specifics is invited to look at the OONI test documentation [26]

5.5 Traffic Manipulation

These are tests which aim to detect the presence of deep packet inspection (DPI) devices. Their aim is not that of detecting what is being censored, but rather if and how. When running a traffic manipulation test, one is trying to answer the question: "If there is surveillance or censorship, where is it likely taking place?" When possible, we will attempt to fingerprint the surveillance and censorship techniques and the specific device or software used.

Tests which fit into this category are:

Latency based measurements This involves noticing timing differences in packets which are directed at specific ports or which possess certain packet structure. If the timing difference between packets is significant, this may be an indication of packet inspection.

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 indicative 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 if traffic has been tampered with on the receiving end.

5.6 Content Blocking

The goal of these tests is to detect what content is being restricted. These tests typically involve iteration through a list of keywords or hostnames in order to identify which subset of these are being blocked. When running a content blocking test, one is trying to answer the question "Is there censorship, and, if so, what is being censored?". When possible, we will attempt to determine the technique used for censoring content and the specific device or software being used.

Examples of tests that are in this category are:

HTTP Host The Host header field of an HTTP request is changed to that of the site one wishes to check for censorship.

DNS lookup A DNS lookup is resolved for a given hostname to verify that it matches the expected result.

Keyword filtering Data containing keywords which are suspected of being censored are sent and received on the test network. The set is then bisected to determine the subsets of keywords which are triggering the filter.

HTTP scan A full connection to the site in question is attempted, and, if the content does not match the expected result, then a censored candidate flag is raised.

Multi-protocol Traceroute TCP, UDP, ICMP, and IP traceroute-style packets are generated for specific destination addresses. If there are discrepancies in the routing paths within the local vicinity of either endpoint, then a censorship candidate flag is raised.

RST packet detection A connection to a certain destination is attempted to check whether or not the response is a TCP RST packet.

CrossBear Implementation of CrossBear [27] [28] to detect and localize SSL/TLS Men-in-the-middle.

6 Data Collection

All data collected by OONI must be open and accessible, and well as consensually submitted by the users. We will make efforts to not redact any data that is contained in the logs, unless it is content could potentially lead to harm or identification of users. We also intend, where possible, for users to submit data over the Tor [29] network, so that they do not connect back to a centralized server, reducing that facet of fingerprint-ability.

We do not consider the data anonymization problem to be solved by any project at this time. Projects either refuse to release their large archive of data or they do not address it meaningfully.

The data format for the reports is YAMLOONI, a document format based on YAML [30]. Every report must contain as a bare minimum the following information:

- Timestamp of start and end of test
- BGP ASN from which the test originated
- The types, kinds and versions of any tests run
- The test results

The timestamp format we use is that specified in RFC3339 [31]. All times are expressed in UTC.

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

When running a test with the assistance of an OONIB, the report from the client is paired with the data collected on the server. This allows us to have both connection viewpoints.

7 Community

In order for the network surveillance and censorship detection field to progress, it is necessary to build a community. We hope by making this software and data available to the public, that a large diversity of people interested in these topics will gather around OONI.

People who are likely to be interested are researchers developing network filtering detection tools, social scientists or legal scholars [23, 22] who want high quality data about censorship around the world, and data visualization specialists who seek to increase public awareness of trends.

Developers will also be interested in OONI, because they will be able to write their own tests with reduced effort. Social scientists will be able to use data collected with OONI as a basis for further research, and for exploration into the political, sociological, and economic contexts of countries where surveillance and censorship are prevalent. Journalists will be able to cite a trustworthy source when they wish to do data journalism. People interested the wider matter of censorship will be able to draw upon the OONI data to make their point.

8 Deployment

The software is created with the intention of being deployed both on servers and on client machines, running Windows, OSX or Linux.

It will also be deployed through integration into the upcoming Torouter [32], allowing people to assist simply by configuring their router to run those OONI tests which they feel comfortable with running.

9 Implementation Status

At the moment of writing of this paper, a prototype of OONI has been implemented. OONI is written in Python and is based upon the Twisted networking framework. The current implementation of OONI provides a framework for test writing, support for locally running tests and a local reporting mechanism. Some parts of OONIB have been implemented, but they do not yet interact properly with OONIProbe. Utility libraries for writing layer 2 packets have been written based upon scapy. These libraries wrap scapy and make sending and receiving functions non-blocking. The packet capture system has been implemented, based on libpcap and libdnet.

10 Limits and future work

We have not discussed a large number of issues inside of this paper because of FOCI size constraints. It is, for example, unclear how much data we should be collecting in test results and how to properly anonymize it; nor have we found a good heuristic for producing categorical lists of content likely to be censored. This needle in a hackstack problem requires human tuning and dovetails nicely with projects such as Herdict [33, 34].

Our threat model is not entirely clear, and we are still unsure if there will be active effort by ISPs 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 OONIProbe. By making it harder for the censor to detect the detection attempt we will probably end up with slightly less reliable results, though this is still not fully clear.

We also did not deal with the implementation specifics for tests that we wish to run with OONIProbe, and the details of censorship detection tests will be the subject of future papers.

References

[1] Philipp Winter. "censorbib: Selected papers in censorship". http://www.cs.kau.se/philwint/censorbib/, 2012. [Online; accessed 28-June-2012].

[2] Richard P. Feynman. *Surely You're Joking Mr. Feynman*. 1985.

[3] Christian Kreibich, Nicholas Weaver, Boris Nechaev, and Vern Paxson. Netalyzr: illuminating the edge network. In Mark Allman, editor, *Internet Measurement Conference*, pages 246–259. ACM, 2010. [Online; accessed 28-June-2012].

[4] Electronic Frontier Foundation. Swizerland. https://www.eff.org/pages/switzerland-network-testing-tool, 2010. [Online; accessed 28-June-2012].

[5] Bennett Haselton. How to use steganography to securely circumvent network-level censorship. http://www.peacefire.org/techpapers/steganography-network-level.html, 2000. [Online; accessed 28-June-2012].

[6] Eddy L O "Saruman" Jansson of DFR Research & Engineering. The penetration of cybersitter'97. http://web.archive.org/web/20050306044647/http://polywog.navpoint.com/reveng/cs97/cs97hack.html, 1997. [Online; accessed 28-June-2012].

[7] Eddy L O "Saruman" Jansson of DFR Research & Engineering. The reversal of netnanny. http://www.polywog.org/reveng/nn/reversal_of_netnanny.html, 1999. [Online; accessed 28-June-2012].

[8] J. Zittrain. Be careful what you ask for: Reconciling a global internet and local law. *Who rules the net*, pages 13–30, 2003.

[9] F. Von Lohmann and Electronic Frontier Foundation. *Unintended Consequences: Twelve Years under the DMCA*. Electronic Frontier Foundation, 2010.

[10] S. Finkelstein. Keeping it clean. *Index on Censorship*, 38(1):123–127, 2009.

[11] S. Cherry. The net effect. *IEEE Spectrum*, 2005.

[12] Andreas Sfakianakis, Elias Athanasopoulos, and Sotiris Ioannidis. CensMon: A Web Censorship Monitor. In *USENIX Workshop on Free and Open Communications on the Internet*, San Francisco, CA, 2011. USENIX Association.

[13] OpenNet Initiative. rturtle. private source code leak and private correspondence, 2009.

[14] OpenNet Initiative. Opennet initiative. http://opennet.net/research/, 2012. [Online; accessed 28-June-2012].

[15] The Tor Project. ooni probe git repository. https://gitweb.torproject.org/ooni-probe.git, 2011. [Online; accessed 28-June-2012].

[16] Creative Commons. Creative commons attribution 3.0 unported license. http://creativecommons.org/licenses/by/3.0/. [Online; accessed 28-June-2012].

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

[18] William Shakespeare. Hamlet: Act 5, scene 2. http://shakespeare-navigators.com/hamlet/HamletNotes52.html#17.

[19] Yale Law School Roundtable on Data and Code Sharing. Reproducible research. *Computing in Science and Engineering*, 12:8–13, 2010. [Online; accessed 28-June-2012].

[20] L. Lessig. What things regulate speech: Cda 2.0 vs. filtering. *Jurimetrics*, 38:629, 1997.

[21] J. Weinberg. Rating the net. *Hastings Comm. & Ent. LJ*, 19:453, 1996.

[22] J.M. Balkin. Media filters, the v-chip, and the foundations of broadcast regulation. *Duke Law Journal*, 45(6):1131–1175, 1996.

[23] R.J. Peltz. Use the filter you were born with: The unconsitutionality of mandatory internet filtering for the adult patrons of public libraries. *Wash. L. Rev.*, 77:397, 2002.

[24] OpenNet Initiative. rturtle. private source code leak and analysis, 2009 - 2012.

[25] The Tor Project. Tor rendezvous specification. https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=rend-spec.txt, 2004.

[26] The Tor Project. Ooni test list. https://trac.torproject.org/projects/tor/wiki/doc/OONI/Tests, 2012.

[27] Ralph Holz, Thomas Riedmaier, Nils Kammenhuber, and Georg Carle. X.509 Forensics: Detecting and Localising the SSL/TLS Men-in-the-middle. In *Proc. 17th European Symposium on Research in Computer Security (ESORICS 2012)*, Pisa, Italy, September 2012.

[28] Crossbear Team (Ralph Holz and Thomas Riedmaier). Turning the tables and how we got there. https://pki.net.in.tum.de/files/berlinsides.pdf. [Online; accessed 28-June-2012].

[29] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[30] Oren Ben-Kiki, Clark Evans, and Ingy döt Net. YAML ain't markup language (YAML) (tm) version 1.2. Technical report, YAML.org, 9 2009. [Online; accessed 28-June-2012].

[31] G. Klyne and C. Newman. Date and Time on the Internet: Timestamps. http://www.ietf.org/rfc/rfc3339.txt, July 2002. RFC 3339 (Proposed Standard).

[32] The Tor Project. Torouter. https://trac.torproject.org/projects/tor/wiki/doc/Torouter, 2012. [Online; accessed 28-June-2012].

[33] Berkman Center for Internet & Society at Harvard University. Herdict. https://www.herdict.org/, 2012. [Online; accessed 28-June-2012].

[34] T. Hwang. Herdict: a distributed model for threats online. *Network Security*, 2007(8):15–18, 2007.