Platform for Teaching Hands-on End-to-End Anonymity Algorithms

Dr. Emil H Salib, Grant Hobar
College of Integrated Science and Engineering
James Madison University
salibeh@jmu.edu, hobargj@dukes.jmu.edu

***Abstract*—This full paper presents an innovative approach in teaching anonymity algorithms. Teaching end-to-end anonymity algorithms, such as onion routing, has been a challenging task. This is primarily because of the inaccessibility of all the components that make up the end-to-end anonymity network. For example, an onion is the data structure formed by "wrapping" a message with successive layers of encryption to be decrypted by as many intermediary computers as there are layers before arriving at its destination. The intent of the onion routing algorithms is to protect the privacy of its users as well as their freedom and ability to conduct confidential communication by keeping their Internet activities from being monitored. In recent years, community research, specifically at the university level, into onion routing has ground to a halt and the limited resources for establishing private testing networks have fallen into a derelict state.**

The proposed solution is an elaborate walk-through, detailing the construction of an isolated onion routing network in which onion routing can be observed and verified in a controlled environment. Achieving this will require the configuration of several Directory Authorities, multiple nodes configured as Entry, Middle, and Exit relays, and the generation of a consensus. Once the consensus has been generated, the network will be able to build circuits across the nodes and fulfill the client's requests for information. When a packet capturing tool is applied to the various entities, the life of the packet as it traverses the network will be observed.

The product of this endeavor was the creation of a complete virtual environment and detailed documentation of the steps necessary to construct a private onion routing network that can be readily utilized by educators and students to teach and learn the innerworkings of the OR algorithms. Constructing an isolated OR network, where all the network traffic is observable, is an essential first step in peeling back the next layer of onion routing.

***Index Terms*—onion routing; Tor; curriculum; teaching method; undergraduate course, anonymity, consensus**

I. Introduction

One of the most popular implementations of the Onion Routing (OR) [1], [2] algorithm is Tor. Tor [3] is a free software for enabling anonymous communication. The name is derived from an acronym for the original software project name "The Onion Router." Tor directs Internet traffic through a free, worldwide, volunteer-overlay network consisting of more than seven (7) thousand relays to prevent people and/or organizations from learning a user's location. It protects against the identification of user's browsing habits and usage from anyone conducting network surveillance or traffic analysis. Using Tor has made it more difficult to trace Internet activity to the user. Tor is intended to protect the personal privacy of its users as well as their freedom and ability to conduct confidential communication by keeping their Internet activities from being monitored.

The core principle of Tor, was developed in the mid-1990s by United States Naval Research Laboratory employees, mathematician Paul Syverson, and computer scientists Michael G. Reed and David Goldschlag, to protect U.S. intelligence communications online [4]. onion routing (OR) was further developed by DARPA in 1997. It is implemented by encryption in the application layer of a communication protocol stack, nested like the layers of an onion [5]. The following three parts on how tor works are excellent introduction to Tor innerworkings [5], [6], [7]. Tor encrypts the data, including the next node destination IP address, multiple times and sends it through a virtual circuit comprising successive, randomly selected Tor relays. Each relay decrypts a layer of encryption to reveal the next relay in the circuit to pass the remaining encrypted data on to it. The final relay decrypts the innermost layer of encryption and sends the original data to its destination without revealing or knowing the source IP address. Because the routing of the communication is partly concealed at every hop in the Tor circuit, this method eliminates any single point at which the communicating peers can be determined through network surveillance that relies upon knowing its source and destination.

The client negotiates a separate set of encryption keys for each hop along the circuit to ensure that each hop can't trace these connections as they pass through. Tor uses its own protocol to negotiate three (3) encryption keys. Through this protocol the client receives encryption keys for Entry/Guard Relay (Routers A's), Middle Relay (Router B's) and Exit Relay (Router C's) [5], [8]. Before a packet is sent, it is encrypted with three encryptions. The client encrypts the original data in such a way that only the Exit Relay can decrypt it. Then, this encrypted data is then encrypted again in such a way that only the Middle Relay can decrypt it. Finally, this encrypted data is encrypted once more in such a way that only the Entry/Guard Relay can decrypt it. For OR to work, a minimum of three (3) routers are required so that no two routers will ever know both the source and destination IP address of the Tor client. When a Tor client sends a packet out, the Entry/Guard Relay will know the source IP address, but not the destination. The Exit Relay will know the end destination, but not the real source

978-1-5386-1174-6/18/$31.00 ©2018 IEEE

address, just the previous relay. The IP address the web server uses for the first return hop will be the Exit Relay's. For more info look at the original Tor design paper [1].

It is clear from the above description of the Tor algorithm implementation that the public anonymity network is not the right environment for educational purposes. To introduce students and educators to the innerworkings of the Tor algorithm, there is a desperate need for an isolated anonymity platform where the students have total control of all the components and their configurations. We have identified a number of efforts towards the implementation of isolated anonymity platforms [9], [10], [11], [12]. However, none of these has been built with a strong focus on teaching undergraduate courses.

II. Solution Design and Implementation

The main goal of this effort is to provide a platform (a set of tools) that assists students and teachers in acquiring hands-on knowledge of the anonymity Tor algorithms, architecture and applications. In this section, we will describe the platform we created and adopted in the implementation of our solution of an isolated Tor anonymity network. Figure 1 depicts the network we have chosen to implement and analyze the Tor algorithm. The network was created by utilizing virtual machines (Ubuntu 16.04-64) and configuring them for respective roles.

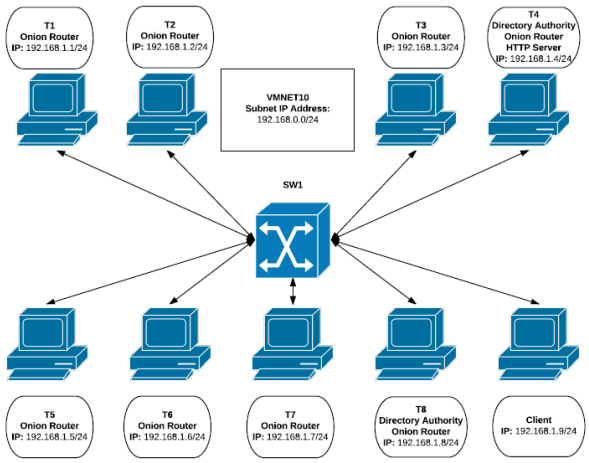


Fig. 1: Network Architecture diagram for lab exercises.

The following is a brief description of the main components and utilities of the solution platform. It is primarily for the benefits of educators interested in adopting our solution platform in teaching the Tor algorithms.

A. Directory Authority

Think of the Directory Authority (DA) servers as the trusted providers of a phone-book. This phone-book, known as the consensus, contains the complete information about each known onion relay and is updated every hour. When it is time to update the list, a majority of the directory authorities must agree on the accuracy of the new list by cryptographically signing the proposed consensus. Once this process is complete, clients are able to download the updated list of relays. How do client know what the DAs are and how to download the consensus from them? The Tor software implementation (e.g., Tor) comes with a built-in list of locations and public keys for each DA. By having trusted authorities keeping a master list of relays and their capabilities, it's straight forward for new and existing clients to keep track of the addition and removal of entities in the anonymity network.

The consensus is generated between DAs in the public anonymity network using the following steps:

- Each DA compiles a list of all the known relays and then computes the other needed data such as relay flags, and bandwidth weights;
- A DA then submits this data as a status-vote to all the other DAs
- After this, each DA will get any other votes it is missing from the other authorities;
- Once all the votes have been gathered, all the relay information from each vote is combined and then signed by each DA;
- This signature is then posted to the other DAs;
- Once a majority of the DAs agree on the data, a new consensus has been established and is then distributed by each DA.

The consensus document itself consists of four main components: Preamble, Authorities, Routers, and Footer. The Preamble includes the technical details about the consensus such as the status, expiration, and software version. The Authorities and Router subsections are almost identical and are where the information regarding each entity is outlined including the identity key, IP address, port configuration, flags, and protocol version. The Footer is reserved for information regarding the bandwidth and signatures. The bandwidth weight section logs the option weights to apply to router bandwidths during path selection. The end of the consensus contains a list of the signatures of DAs in the format of a hex-encoded digest of the identity key of the signing authority.

During the creation of a DA, a number of Authority Keys are generated: the first key, authority_identity_key, is your long-term master identity key that is used to sign authority certificates; the second key, authority_signing_key, is the medium-term key that is used to sign directory information such as votes and consensuses; and the third key, authority_certificate, is a document used to authenticate the authoritys current vote and is signed by the authority_identity_key to certify the authority_signing_key.

The purpose of each key is as follows: the secret_id_key file is the relays RSA1024 permanent identity key, including private and public components (used to sign the router descriptor and TLS certificates); the secret_onion_key file is the medium-term RSA1024 key used to establish a circuit and negotiate ephemeral keys; and the secret_onion_key_ntor file is the short-term key and is primarily used for handshakes that involve anything circuit-extension related. The fingerprint file is the fingerprint of the identity key.

In the real-world, there are 10 DAs whose information is hard-coded into the Tor Clients. These gatekeepers have been chosen by the developers because they are dependable and trustworthy. Typically, it is a bad idea to have an even number of entities in a voting body because it could result in a tie, however the Tor network actually employs nine (9) DAs to maintain the master list of relays, while a single DA is tasked with maintaining the list of bridges.

B. Relays

Relays are the backbone of the Tor network. They establish and pass messages along circuits. Each relay is a volunteer who donates the bandwidth to the Tor network by advertising themselves to the DAs. There are three types of Tor relays: Entry/Guard, Middle, and Exit. When configuring a relay, users can choose between being a Middle relay or an Exit relay Entry/Guard relays are flags that are awarded to relays that have proven themselves to be reliable and trustworthy. Entry/Guard relays have their name and requirement for established trust because they are the first hop in a circuit so they are able to see the Tor clients true IP address. Middle relays are the most popular as they simply receive traffic and pass it along to the next relay in the circuit. An Exit relay is the final relay that Tor traffic passes through before it reaches its destination. Because Tor traffic exists through these relays, the IP address of the Exit relay is interpreted as the source of the traffic therefore masking the clients request. Here, the content of the message is no longer encrypted in the Tor protocol. They have been peeled away to their core and can now be routed to their true destination. If the DA detects in the consensus that there are no Guard or Exit relays, yet there is an abundance of Middle relays, it will then attempt to designate them as such by determining a variety of factors such as their bandwidth and age. The longer a relay has been online and the more traffic it can handle the more trusted it becomes to the DA.

During the creation of an Tor relay, a number of Relay Keys are generated in the process, namely - secret_id_key, secret_onion_key, and secret_onion_key_ntor. The secret_id_key file is the relays RSA 1024 permanent identity key, including private and public components (used to sign the router descriptor and TLS certificate.) The secret_onion_key file is the medium-term RSA 1024 key used to establish a circuit and negotiate ephemeral keys. The secret_onion_key_ntor file is the short-term key and is primarily used for handshakes that involve anything circuit-extension related.

When an Tor client starts up, it needs a way to fetch the list of all the entry, middle, and exit relays available. As previously mentioned, this list is known as the Consensus. Having this list be accessible by the public is necessary to the Tor core function, however it also presents a major problem. To figure out why this is a problem, one must realize that the Tor network is largely utilized for browsing Internet content anonymously and circumventing censorship. Government agencies and organizations that implement this censorship obviously want it to remain effective so they have decided to attempt to block Tor users. There are only two ways to do this: blocking users leaving Tor, and blocking users entering Tor. The first scenario is easily accomplished at the discretion of the network administrator by accessing the consensus to view the list of exit relays. Once they have downloaded this list, all they would have to do is block any traffic from those nodes or relays. While blocking incoming Tor users can prevent them from accessing a particular website, blocking users from entering the Tor network will keep them from accessing every website, making OR effectively useless to those under censorship. Thankfully, the Tor project thought of exactly this situation and came up with a solution known as Bridges.

Bridges are a clever solution to the problem posed by organizations attempting to block users from utilizing the Tor network. At their core, Bridges are simply unpublished entry relays. Users that are behind censored networks can use these Bridges to access the Tor network. So if Bridges are unpublished, how do users know where they are? Wont a master list need to be published somewhere? Simply put, the information about the Bridges is obtained the same way that clients receive the data about DAs, this list is just not made public. As previously mentioned, of the ten (10) DAs in the public network only one of them handles the list of bridges. By only distributing a few Bridges at a time, it is possible to prevent organizations from blocking all possible entry points in the Tor network. To keep the Tor network as anonymous and secure as possible, it's designed to intentionally distrust relay operators.

C. Client, HTTP Web Server, and Utilities

OR does not protect all of a computer's Internet traffic when run. It only protects applications that are properly configured to send their Internet traffic through OR. To avoid problems with Tor configuration, the Tor project developers strongly recommend the use of the Tor Browser Bundle. It is preconfigured to protect privacy and anonymity on the web as long as browsing is done with the Tor Browser itself. Almost any other web browser configuration is likely to be unsafe to use with Tor.

OR provides only anonymity for DNS and the transmission of the TCP stream. Everything inside the stream, such as the Application-layer protocol, needs to be scrubbed. If an application uses advanced techniques to determine a real external IP and sends it over the anonymized TCP stream, that will compromise anonymity and reveal the true IP address. An example of an application that was considered to be anonymous, but turned out to be doing what was just described, is Bit-Torrent. Some applications may also choose to ignore and therefore not honor the proxy configuration provided. Firefox was prone to this issue, as noted in [13].

Certain software identifiers can also expose anonymity while traversing the Tor network. You must be very careful when using any software, particularly proprietary that is activated with some type of a license or a key. Various governments and private organizations have proven it to be possible to explicitly or steganographically insert hardware and

environment identifiers into documents to be able to track and profile users or using the software for illegal purposes.

To "Torify" an application, means to take measures to ensure that an application, which has not been designed for use with Tor routing (e.g., Tor), will use only Tor for Internet connectivity - ensuring that there are no leaks from DNS, UDP, or the Application protocol. Once the Tor software has been installed on a machine it is possible to manually configure a proxy server in the network settings of a web browser to be directed your Internet traffic towards the port that the Tor service is listening to. By Torifying a web browser, all of your Internet traffic will be passed through the Tor network, creating circuits with various relays in the network on-demand. In an isolated network it is necessary to create an HTTP web server that is accessible by all of the clients within the network, otherwise there will be no content to access - meaning that there would be nothing to analyze!

Several Python utility scripts were written over the course of this project to assist in troubleshooting and analyzing the Tor network. These scripts relied upon the STEM Python controller library [14] that was designed for interacting with Tor processes. The first program created was called *Circuit Trace*, designed to record the circuit number, the fingerprint of each relay involved in the circuit, and the relay's name and IP address as well as the order of the relays in ascending order from entry guard to exit relay. The second program created was *Exit Trace*, which functions similarly to Circuit Trace, however this script only returned the exit relay of the current circuit and various technical information about the relay. This was predominately designed to act as a redundancy check for Circuit Trace. The final program was called *Scrubber* and was a utility for quickly clearing cached information, such as the consensus and logs, as well as even keys and fingerprints.

The last utility for analyzing Tor network was Wireshark. This is a free, open-source packet analyzer used for network troubleshooting, analysis, and education. To verify the claims that the Tor network claims regarding OR, it was imperative to visualize the packet exchanges occurring between the entities of the network - particularly the relays involved in a circuit.

III. Methods & Results

A. Exercise 1 : Configuration of the Isolated Environment

a) Introduction:

The need for creating an isolated network, as opposed to utilizing the existing public network, stems from the requirements to have control all of the entities involved in the network as well as having the ability to monitor the packet flow between them. This approach allows the students to both visualize and verify the functionality of the Tor protocols. The network architecture for this experiment was chosen to emulate the Tor network with the smallest amount of actors necessary to observe the Tor protocol. The network consists of eight relays:six of which are dedicated relays, and two are DAs because they can also act as entry guards or exit relays if there are no suitable candidates. One of the DAs also hosts an HTTP web server that any actor within the network will be able to access. This could be hosted on a dedicated node, however for this project it was combined. Two DAs were chosen so that a consensus could be generated, because it was determined through many trials that the consensus would not be generated if there was only one DA in the network. The students will be asked to investigate this issue through the same processes and steps. That is, they will be asked to start with one DA and fewer relays. Such q configuration may work occasionally, but it will be consistently unreliable. They will then be asked to explore the concept of consensus and pondering whether one DA is meaningful and practical to meet the needs of reaching a consensus. The final actor involved in the network architecture is the Client. This will be the individual triggering the generation of circuits within the network and allowing the user to trace each hop the packet takes throughout the circuit.

b) Public Tor Network Limitations:

When accessing any Internet resources, such as an HTTP web server, through either a Torified application or the official Tor browser, a circuit will be established as shown in Figure 2. As previously mentioned, the circuit consists of three main components: Entry Guard, Middle Relay, and Exit Relay. Any data request being transmitted through the Tor network will be encrypted via the Tor protocol from the point the message leaves the Client all the way to the Exit Relay where the data request is decrypted and executed by the Exit Relay before reversing back to the client.

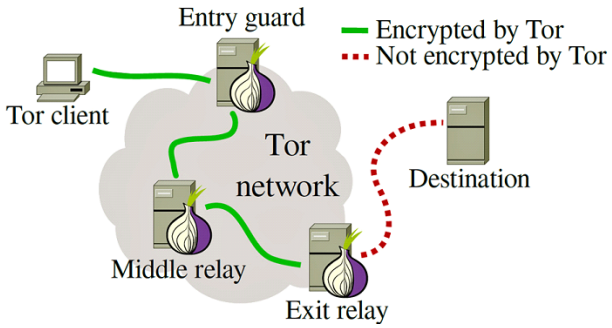


Fig. 2: Visual illustration of the path that a message takes while traversing the Tor network [15].

Only the official Tor Browser can observe the circuit involved in the communications with individual web applications. One of the ways that students may verify that their traffic is being masked is to verify the public IP address associated with a data request. If the public IP matches the IP of the exit relay involved in the circuit, then the Tor circuit is functioning correctly. While it is excellent to verify that the exit relay is seemingly operational, there is no way to observe the packet exchange between any of the intermediary relays in the circuit: the Tor protocol can't be verified. An isolated Tor network however remedies this problem because the students control all the actors and can monitor the network traffic between them. Another advantage that the isolated network provides is in regards to the consensus. The public Tor network's consensus is accessible by anyone, however with

such a sprawling network it is impossible to decipher and correlate data. With a small isolated network the consensus will be concise, making it easier to read, while also giving the user the ability to compare each fingerprint and relay descriptor with the actual entity. This is where the students will be asked to compare the potential difference in performance between the isolated, limited in scope network versus the public network. In a public network, traffic may traverse a number of countries (in different continents) before reaching the other end as shown in Figure 3. The only way for the

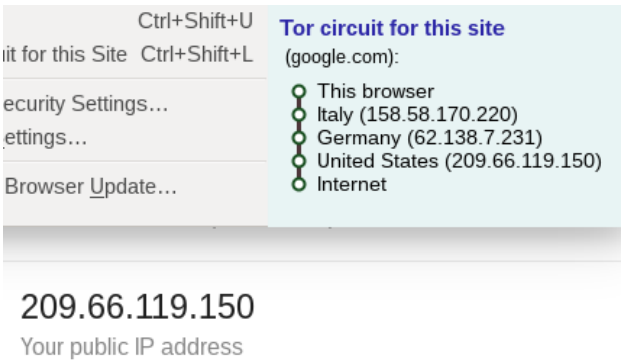


Fig. 3: Utilizing the official Tor Browser to view the Tor circuit established for the website

students to view active circuits in the Tor network other than through the Tor Browser is to participate in the network and configure a middle relay. By utilizing a built-in command-line monitoring application for Tor [16] it is possible to get detailed real-time information regarding their relay such as bandwidth consumption, active connections, and event logs. The monitoring application allows the students to observe each individual circuit that the relay is involved with as well as more technical information such as the IP addresses, names, and functions of the other relays involved. Since relays take weeks of stable performance to become trusted and earn either a guard or exit flag, it will always appear as the Middle relay, leaving the students' relays unexposed to two-thirds of the types of Tor relays as demonstrated in Figure 4. However, it

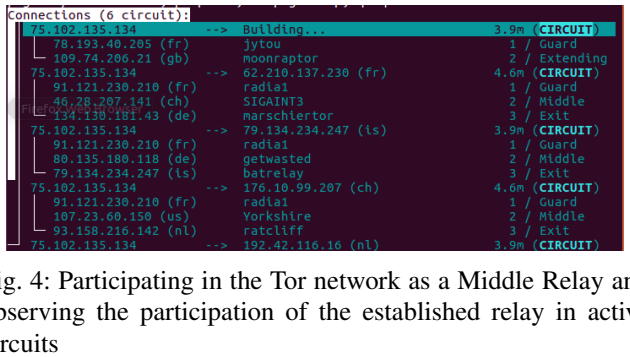


Fig. 4: Participating in the Tor network as a Middle Relay and observing the participation of the established relay in active circuits

is imperative that the students be able to observe all of the involved actors of the network to fully understand the role each one performs. This can be realized through an isolated Tor Network.

c) Isolated Tor Network Configuration:
The configuration of an isolated Tor network is quite a cumbersome task. It will stretch the students' network troubleshooting capabilities and require them to bridge conceptual gaps to advance through the steps necessary to configure all of the actors in the network. By configuring each component the students will understand how each actor is related to one another and verify the functionality of each one. This part of the exercise relies upon the usage of several Linux virtual machines all configured within the same network. All of the machines will have the same core Tor software package installed on them as well as Wireshark for monitoring the network communications between the actors. Each machine has its own unique fingerprint associated with it as well as a set of signing keys. These public fingerprints will be used to identify actors within the consensus. There are several types of configuration profiles that need to be implemented within the Tor source code, each profile representing an actor: DA, Relay, and Client. The DA configuration profile is the most complex out of the three primarily due to the additional syntax required for consensus generation and communication with other DAs. One of the DAs also must be configured with an HTTP web server that is accessible by everyone within the network. This step is necessary because the students will be asked to generate a circuit where the data must be requested by the Client first. Because this is an isolated network there is no other content to access. The configuration profile for the Relay is similar to the DAs, but shorter since they do not have to deal with the consensus generation. Because six relays are configured in the network we want half of the relays to have DA1 as their primary DA, and the other half with DA2 as their primary. The importance of this would be discovered by the students in the consensus generation phase when each DA exchanges its individual list of relays for the first time and DA2 uses relays in DA1's "domain" and vice versa. Also, the students will be able to observe and analyze the packet exchange during the introduction process between the individual relays and the DAs. The Client configuration profile is quite simple and essentially relies solely upon pointing towards the primary DA. However, up to this point, the Tor network has not been initialized and is still in a state where the machines should all be able to communicate over the network as well as have the ability to access the web page that was configured.

d) Launching the Isolated Tor Network:
Once all of the fingerprints and keys have been generated, IP addresses have been manually set, and network connectivity between actors has been established, then students will be able to launch the Isolated Tor Network. Through much experimentation, the students will be asked to determine that the order in which they launch each Tor application is critical. The DAs must be the first Tor entities on the network because of their role as the gatekeeper of the consensus. When establishing the Tor Network, if there is a relay older than the first DA the relay is flagged as a result, meaning that it

will receive no traffic and not be counted in the consensus of valid relays. When the DA is the first entity in the network it is able to populate the consensus as the Tor relays reach out to the DA with handshakes containing their fingerprints to identify themselves to the server as well as request the current consensus to start developing circuits between actors. Following the establishment of the DAs, the students will be asked to proceed to launch the relays. The order in which they launch the relays will actually dictate their eligibility for flags such as Guard and Exit. Once all of the relays have been established, the network requires some time to generate a consensus between the two Directory Authorities. While the consensus is being generated, the Client can be brought online. Once a circuit has been established the students should be able to access the HTTP web server.

Utilizing the Python utilities developed through the process of researching this project and introduced in section II, the students will be able to verify the exact path the network took to reach the HTTP web page. Figure 5 shows an example of

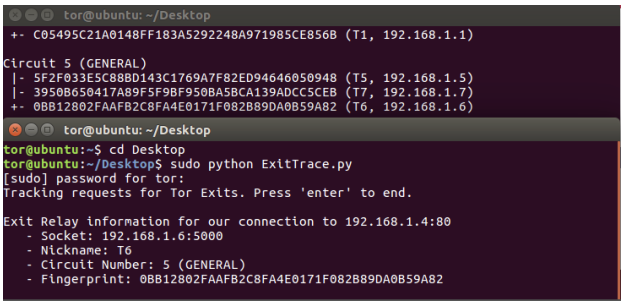


Fig. 5: Successfully establishing a Tor Circuit from the Client across the Isolated Tor Network to access the HTTP web server

what the student would expect. The Client uses Circuit 5 to access the HTTP web server hosted on T4. The entry relay was ”T5” (192.168.1.5), the Middle Relay was ”T7” (192.168.1.7), and the Exit Relay was ”T6” (192.168.1.6).

As previously mentioned, the relays that were booted earlier than the others have a higher chance of being assigned a Guard or Exit flag. The main factors that go into determining Entry Guards and Exit relays are the length of the up-time and the amount of bandwidth provide. This is where the students will be asked to conduct a number of experiments to determine the parameters and criteria involved in controlling the selection processes. If a relay is consistently online with minimal down-time and offers high speeds, then it will gradually become trusted and earn the respective flags. According to the Tor Project, there are four main phases: Unmeasured (Days 0-3), Remote-Measured (Days 3-8), Ramp-Up (Days 8-68), and Steady-State (Days 68+). Each one of these phases performs several benchmark tests, ensuring that the relay is in fact good and can be trusted. Students simply do not have the time to wait two months for a relay to be considered steady-state as in the public network, so in the Isolated Tor Network the time is drastically reduced to be few minutes or even a few seconds in some cases. With the network now emulating real-world conditions, students should have a solid grasp of the fundamental concepts that go into the structure and data flow of the Tor network, as well as begin to have a clear view of how the Tor protocol is observable in this network.

B. Exercise 2: Isolated Tor Network Analysis

a) Introduction:

This exercise focuses on analyzing the generation of circuits and the consensus. The generation of circuits is a core function of the Tor network and this environment allows students to visualize handshakes between relays and the incremental assembly of circuits by the Client as well as verify the Tor Protocol through packet analysis. The students will develop the ability to decipher the consensus as well the generation and distribution of the document to the network. Analyzing the consensus and circuits, coupled with manually constructing the Isolated Tor network should provide students with validation of the Tor Protocol and mastery of the innerworkings of the Tor network.

b) Relay Inter-Communications / Circuit Generation:

Before a circuit may be established between relays, the relays must first introduce themselves to each other through a TLS handshake. This is a simple check to authenticate each relay using the information the DA has provided within the consensus. The relays will authenticate the fingerprints they receive from the other relays using RSA and Ed25519. Once the first relay has authenticated the second relay, it sends an AUTH CHALLENGE to the first relay. Once both actors have successfully authenticated each other a connection is established between the two and they are ready to establish circuits with each other.

The status messages seen in Figure 6 can be found in the log files of the relays and clients. A TLS connection to the

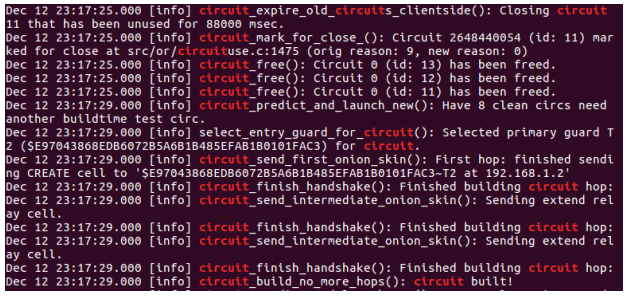


Fig. 6: Circuit generation on the Client

Entry Guard (T1) is established, TLS-T1. Through TLS-T1, the client does a circuit-level handshake to set up shared keys with T1 for the forward and backward paths, KF-T1 and KB-T1, respectively. The client next creates a RELAY EXTEND cell to extend the circuit to the Middle relay (T2) which contains the first stage of the circuit-level handshake with T2. It encrypts this relay cell with KF-T1 and sends it forward to T1, which decrypts with KF-T1 and packages the content into a RELAY CREATE cell, which is sent over a newly established TLS connection between T1 and T2. TLS-T2 sends its half of the circuit handshake in response, packaged

in a RELAY CREATE cell and reverse encrypted (with the corresponding KBi keys) down the reverse path. After the Client's handshake with the Middle relay (T2) completes, the client creates another RELAY EXTEND cell to extend the circuit to the Exit relay, T3. This is then tunneled over TLS-T2, which is tunneled through TLS-T1. The cell itself is super-encrypted with Enc(KF-T2,Enc(KF-T1,CELL)). This cell is sent to T1, who decrypts with KF-T1 and sends it along to T2. T2 decrypts with KF-T2, sees that it's a RELAY EXTEND cell to T3, packages the content into a RELAY CREATE cell, as T1 did before, and sends it to T3. The Exit relay receives this RELAY CREATE cell, performs Dec(KF-T3,CELL) and receives the traffic the client had intended to proxy [17]. At this point, the students will be challenged to figure a way to decrypt the tunnels using Wireshark or modifying the relay code and configurations.

The aforementioned Python utilities can be used in conjunction with the log files and Wireshark captures to observe and verify the creation of a circuit through the Tor network. The *Circuit Trace* utility allows the student to see the exact circuit utilized: including the fingerprint and role each relay played in the circuit. The fingerprint allows the user to verify the identity of the relay and if the Exit relay is known, then the messages exchanged during the deployment of the Tor algorithm will be identifiable in Wireshark.

The Tor circuit that was established in Figure 7 was Circuit number 82. The Entry Guard was relay T5 whose IP address was 192.168.1.5. The Middle relay was T3 whose IP address was 192.168.1.3. The Exit relay was T2 whose IP address was 192.168.1.2. With this information and the Wireless capture

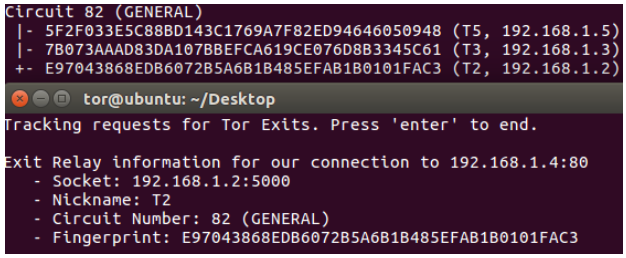
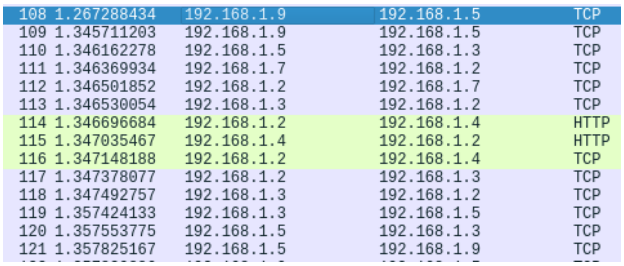


Fig. 7: Python utilities showing the circuit information when connecting to the HTTP web server in the Isolated Tor network

that occurred while accessing the HTTP web page from the Client, the student should be able to identify the exchange of messages that occurred between each actor involved in the circuit. You can see the Wireshark capture of the aforementioned Tor Circuit in Figure 8. The Client (192.168.1.9), initiates the request to the Entry Guard (192.168.1.5), which then passes the request to the Middle relay (192.168.1.3), which peels back another layer and passes the message to the Exit relay (192.168.1.2). Up until now, all of the communication has been secured and viewable only as TCP packets. When the Exit Relay passes the request for the URL to the HTTP web server (192.168.1.4), we are able to see the HTTP GET request before the message is passed back to the Client. All of the IP



108	1.267288434	192.168.1.9	192.168.1.5	TCP
109	1.345711203	192.168.1.9	192.168.1.5	TCP
110	1.346162278	192.168.1.5	192.168.1.3	TCP
111	1.346369934	192.168.1.7	192.168.1.2	TCP
112	1.346501852	192.168.1.2	192.168.1.7	TCP
113	1.346530054	192.168.1.3	192.168.1.2	TCP
114	1.346696684	192.168.1.2	192.168.1.4	HTTP
115	1.347035467	192.168.1.4	192.168.1.2	HTTP
116	1.347148188	192.168.1.2	192.168.1.4	TCP
117	1.347378077	192.168.1.2	192.168.1.3	TCP
118	1.347492757	192.168.1.3	192.168.1.2	TCP
119	1.357424133	192.168.1.3	192.168.1.5	TCP
120	1.357553775	192.168.1.5	192.168.1.3	TCP
121	1.357825167	192.168.1.5	192.168.1.9	TCP

Fig. 8: Wireshark capture on the Client, showing the Circuit used to establish a connection with the HTTP web server

addresses and fingerprints can be cross-referenced and verified in Table 1.

TABLE 1: Key Network Elements Parameters

Nickname	IP Address	EUI-48	Netmask	Fingerprint
T1	192.168.1.1	00:0C:29:99:ac:69	255.255.255.0	C05495C21A0148FF183A5292248A971985CE856B
T2	192.168.1.2	00:0C:29:74:3fe2	255.255.255.0	E97043868EDB6072B5A6B1B485EFAB1B0101FAC3
T3	192.168.1.3	00:0C:29:94:a2:d5	255.255.255.0	7B073AAAD83DA107BBEFCA619CE076D8B3345C61
T4	192.168.1.4	00:0C:29:bd:8a:60	255.255.255.0	87C6C7517B102D1D0DF198DB5F3776DE40061AD8
T5	192.168.1.5	00:0C:29:e8:93:b8	255.255.255.0	5F2F033E5C88BD143C1769A7F82ED94646050948
T6	192.168.1.6	00:0C:29:49:c7:11	255.255.255.0	0BB12802FAAFB2C8FA4E0171F082B89DA0B59A82
T7	192.168.1.7	00:0C:29:13:36:15	255.255.255.0	3950B650417A89F5F9BF950BA5BCA139ADCC5CEB
T8	192.168.1.8	00:0C:29:95:10:d8	255.255.255.0	CB3C5EFE822AB9631A52FEF48FD8EBFB9F872DDD
Client	192.168.1.9	00:0C:29:59:e1:bc	255.255.255.0	67BEA4B5B57CD8EC7E676ADCDB27EE8594FB7AAE

c) Consensus Polling and Distribution:

For DAs to establish a consensus they must first establish a "Trusted" flag with the other DAs. The configuration profile of the DA required the students to enter the fingerprint of the other DA - the DAs perform a handshake similar to the relays and if the fingerprint matches then they are flagged as trusted. By utilizing the log files of the DA, students will be able to discover if and when the DAs handshake occurred, as seen in Figure 9. Earlier in the configuration half of the relays were configured to DA1 as their primary DA and the other half were configured to DA2. This was intentionally done so that the DAs would have to communicate this information via the consensus. The students will be asked to explore the impact on the network performance and behavior for other cases. In Figure 10, the students will see that DA1 learned about the three relays that DA2 has knowledge of via the consensus. In addition to learning about the handshakes that

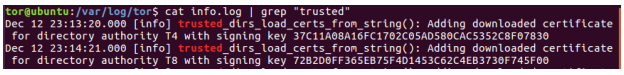


Fig. 9: Log file of one of the DAs indicating the occurrence of the handshake and the two DAs are "trusted"

Fig. 10: Directory Authority 1 is learning about the relays that Directory Authority 2 has knowledge of through the consensus

occur between the relays and DAs, the log files of the DAs also provide information regarding the current status of the consensus. Figure 11 shows that it is ”Time to Vote,” then proceeds to ”Vote Posted,” and then proceeds to ”Fetch any votes that we’re missing.” Once a signature has been fetched from the other DA, it is ”Time to compute a consensus”. The process repeatedly occurs and the students should occasionally see when a DA blocks an actor from the network, resulting in them being ignored by network activity. The log files are an invaluable resource for the students when they are troubleshooting any issues within the Tor network. The students should conclude that most problems can be resolved by examining the logs found on the DA and Clients. One of the most common problems that the students will face is that the DA determines that there is a lack of trusted relays due to there being so few in this small-scale network. The DA will eventually fill the role itself to complete circuits until an eligible relay is available. The alternative to this is forcing the DA to vote for a certain relay as a trusted entity. By vouching for a relay, a DA can effectively force which relays are the designated Entry Guards and Exit Relays.

```
[notice] Time to vote.
[info] dirserv_compute_performance_thresholds(): Cutoffs: For Stab
, 4365 sec MTBF. For Fast: 0 kilobytes/sec. For Guard: WFU 98.000%
and bandwidth 0 or 0 kilobytes/sec. We have enough stability data.
[notice] Choosing valid-after time in vote as 2017-12-13 01:10:00:
st_interval=300
[info] update_consensus_router_descriptor_downloads(): 0 router de
le. 0 delayed; 7 present (0 of those were in old_routers); 0 would
se; 0 in progress.
[notice] Vote posted.
[notice] Uploaded a vote to dirserver 192.168.1.8:7000
[notice] Time to fetch any votes that we're missing.
[notice] Got a signature from 192.168.1.8. Queuing it for the next

[notice] Time to compute a consensus.
[info] networkstatus_compute_consensus(): Generating consensus usi

[notice] Computed bandwidth weights for Case 2b1 (Wgg=weight_scale
 G=1 M=1 E=1 D=1 T=4
[notice] Bandwidth-weight Case 1 is verified and valid.
[info] networkstatus_compute_consensus(): Generating consensus usi

[notice] Computed bandwidth weights for Case 2b1 (Wgg=weight_scale
 G=1 M=1 E=1 D=1 T=4
[notice] Bandwidth-weight Case 1 is verified and valid.
[info] dirvote_add_signatures_to_pending_consensus(): Have 1 signa
ns consensus.
[info] networkstatus_add_detached_signatures(): Looking at signatu
381FC7F9B2C4186414BE6B759D38 using sha1
[info] networkstatus_add_detached_signatures(): Adding signature f
C7F9B2C4186414BE6B759D38 with sha1
[notice] Added a signature for RS8 from pending.
```

Fig. 11: Debug log of Directory Authority indicating the various processes of the consensus generation process

The final step is for the students to analyze the generated consensus document. As previously mentioned, the consensus document itself consists of four main components: Preamble, Authorities, Routers, and Footer. The Preamble includes the technical details about the consensus such as the status, expiration, and software version. The Authorities and Router subsections are almost identical and are where the students will find information regarding each entity outlined including the identity key, IP address, port configuration, flags, and protocol version. The Footer is reserved for information regarding the bandwidth and signatures. The bandwidth weight section logs the option weights to apply to router bandwidths during path selection. The end of the consensus contains a list of the signatures of DAs in the format of a hex-encoded digest of the identity key of the signing authority. Figure 12 shows how to translate the consensus for DAs and Relays.

Fig. 12: Consensus Authorities and Relays Translation

Using the translation guide in Figure 12, the consensus in Figure 13 quickly becomes decipherable. The first relay’s nickname is T6 (it has Exit, Fast, and Stable flags assigned), it is running Tor v3.3.3, and has an IP address of 192.168.1.6. The second relay’s nickname is T7, has Exit and Fast flags, is running Tor v3.3.3, and has an IP address of 192.168.1.7. Every actor within the network other than the Client’s are listed in the consensus. Using the information found in the consensus, coupled with the plethora of log files available, student’s should be able to identify the interactions between all involved actors of the Tor Isolated Network.

```
# Relays
r T6 C7EoAvqvssj6TgFx8IK4naC1moI VcLnNQU4rGTdB9JFE+1Ds3HxGLo 2017-12-13 07:13:52 192.168.1.6 5000 0
s Exit Fast HSDir Running Stable V2Dir Valid
v Tor 0.3.3.0-alpha-dev
pr Cons=1-2 Desc=1-2 DirCache=1-2 HSDir=1-2 HSIntro=3-4 HSRend=1-2 Link=1-4 LinkAuth=1,3 Microdesc=1-2 Relay=1-2
w Bandwidth=0 Unmeasured=1
p reject 25,119,135-139,445,563,1214,4661-4666,6346-6429,6699,6881-6999

r T7 OVC2UEF6ifXSvSULpbyhOa3MXOs yHqwWDli3nlD/C+sTa64Y0gRVNU 2017-12-13 20:38:32 192.168.1.7 5000 0
s Exit Fast Running V2Dir Valid
v Tor 0.3.3.0-alpha-dev
pr Cons=1-2 Desc=1-2 DirCache=1-2 HSDir=1-2 HSIntro=3-4 HSRend=1-2 Link=1-4 LinkAuth=1,3 Microdesc=1-2 Relay=1-2
w Bandwidth=54 Unmeasured=1
p reject 25,119,135-139,445,563,1214,4661-4666,6346-6429,6699,6881-6999
```

Fig. 13: Example of the Relay list generated by the consensus

IV. Conclusions & Next Steps

It is exciting to share our experience of the Tor platform, the utilities we adopted in creating an architecture and designing effective and practical hands-on lab exercises. We believe that these would be valuable to those who are interested in learning and/or teaching the innerworkings of the Tor algorithms. Below is a list of anonymity-related topics that we see ourselves pursuing in hands-on research projects and teaching lab exercises in the near future.

- Configure the Tor implementation to enable students to dive deep into the cryptographic techniques applied
- Expand the solution platform to include the Garlic Routing (e.g., I2P) algorithm [18]
- Expand upon network topology to include more Relays, and DAs to more accurately simulate a real-world environment
- Continue to assess the effectiveness of the platform as a hands-on tool in teaching anonymity algorithms

V. Acknowledgment

The authors would like to thank the Department of Integrated Science and Technology (ISAT) at James Madison University for supporting this effort.

References

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, ser. SSYM'04. Berkeley, CA, USA: USENIX Association, 2004, pp. 21–21. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251375.1251396

[2] R. Dingledine and N. Mathewson, "Tor's protocol specifications," https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt, Retrieved Apr 18, 2018.

[3] "Tor Project Web Site," https://www.torproject.org/, Retrieved: April 2018.

[4] D. Goldschlag, M. Reed, and P. Syverson, "Onion routing," *Commun. ACM*, vol. 42, no. 2, pp. 39–41, Feb. 1999. [Online]. Available: http://doi.acm.org/10.1145/293411.293443

[5] J. Wright, "How Tor Works: Part One," "https://jordan-wright.com/blog/2015/02/28/how-tor-works-part-one/", Feb 28, 2015.

[6] J. Wright, "How tor works: Part two - relays vs. bridges," "https://jordan-wright.com/blog/2015/05/09/how-tor-works-part-two-relays-vs-bridges/", May 9, 2015.

[7] J. Wright, "How tor works part three - the consensus," "https://jordan-wright.com/blog/2015/05/14/how-tor-works-part-three-the-consensus/", May 14, 2015.

[8] "About onion packet and onion routing?" https://security.stackexchange.com/questions/76438/about-onion-packet-and-onion-routing/, Dec 14, 2014.

[9] F. Liu, "How to Setup Private Tor Network," http://fengy.me/prog/2015/01/09/private-tor-network/, Jan 9, 2015.

[10] "private tor network," https://github.com/antitree/private-tor-network, Jun 2, 2017.

[11] A. M. Smith, "private-tor-network-kube," https://github.com/andrewmichaelsmith/private-tor-network-kube, June 2, 2017.

[12] Tracey, Justin, "Building a better tor experimentation platform from the magic of dynamic elfs," Master's thesis, University of Waterloo, 2017.

[13] "Firefox security bug (proxy-bypass) in current TBBs," https://blog.torproject.org/firefox-security-bug-proxy-bypass-current-tbbs, May 2, 2012.

[14] "Welcome to Stem!" https://stem.torproject.org/, Retrieved May 2, 2018.

[15] "Over 100 snooping Tor Nodes have been spying on Dark Web Sites," http://www.securitynewspaper.com/2016/07/02/100-snooping-tor-nodes-spying-dark-web-sites/, July 2, 2016.

[16] "Tor-stable manual," https://www.torproject.org/docs/tor-manual.html.en, Retrieved May 2, 2018.

[17] I. A. Lovecruft, "Tor's Circuit-Layer Cryptography, Attacks, Hacks, and Improvements," https://people.torproject.org/~isis/slides/2016-10-13-waterloo-handout.pdf, October 13th, 2016.

[18] "THE INVISIBLE INTERNET PROJECT," https://geti2p.net/en/, Retrieved: June 2018.