USING TRAFFIC ANALYSIS TO IDENTIFY THE SECOND GENERATION ONION ROUTER

A proposal for a thesis submitted in partial fulfilment of the requirements for the degree of

Computer Science (Honours)

By:	John Barker
Student ID:	0991300
Email:	jebarker@our.ecu.edu.au

School of Computer and Security Science

Edith Cowan University

Supervisors:	Dr Andrew Woodward
	Mr Peter Hannay
	Mr Patryk Szewczyk

Date of submission: October 24, 2010

1 Introduction

1.1 Background

This document describes the procedures for configuring a small network of computer systems to emulate a small sample of web based traffic using both the HTTPS protocol and the Tor protocol. All traffic between the systems is captured for the purpose of traffic analysis.

1.2 Hardware Configuration

All systems will be configured and connected to the same isolated network through the hub as shown in Figure 1.1.

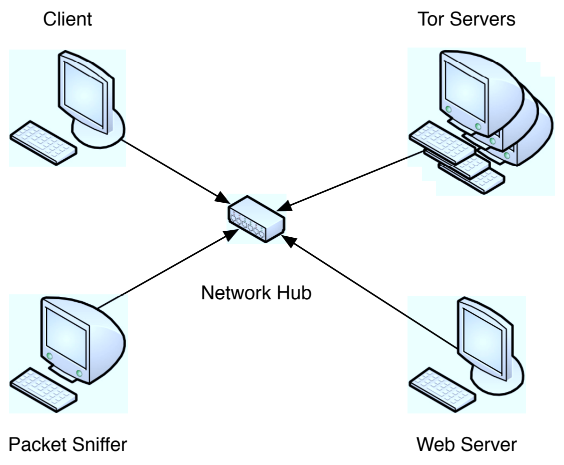

Figure 1.1: Physical Setup

1.3 Software Configuration

Software is installed and configured on all machines. The simulation network is configured to use the private Tor directory server rather than the public ones specified by default (Lewman & Harris, 2009).

The packet sniffer is configured to capture only the packets relevant for each data set.

Figure 1.2 shows the placement of applications and relevant data flows.

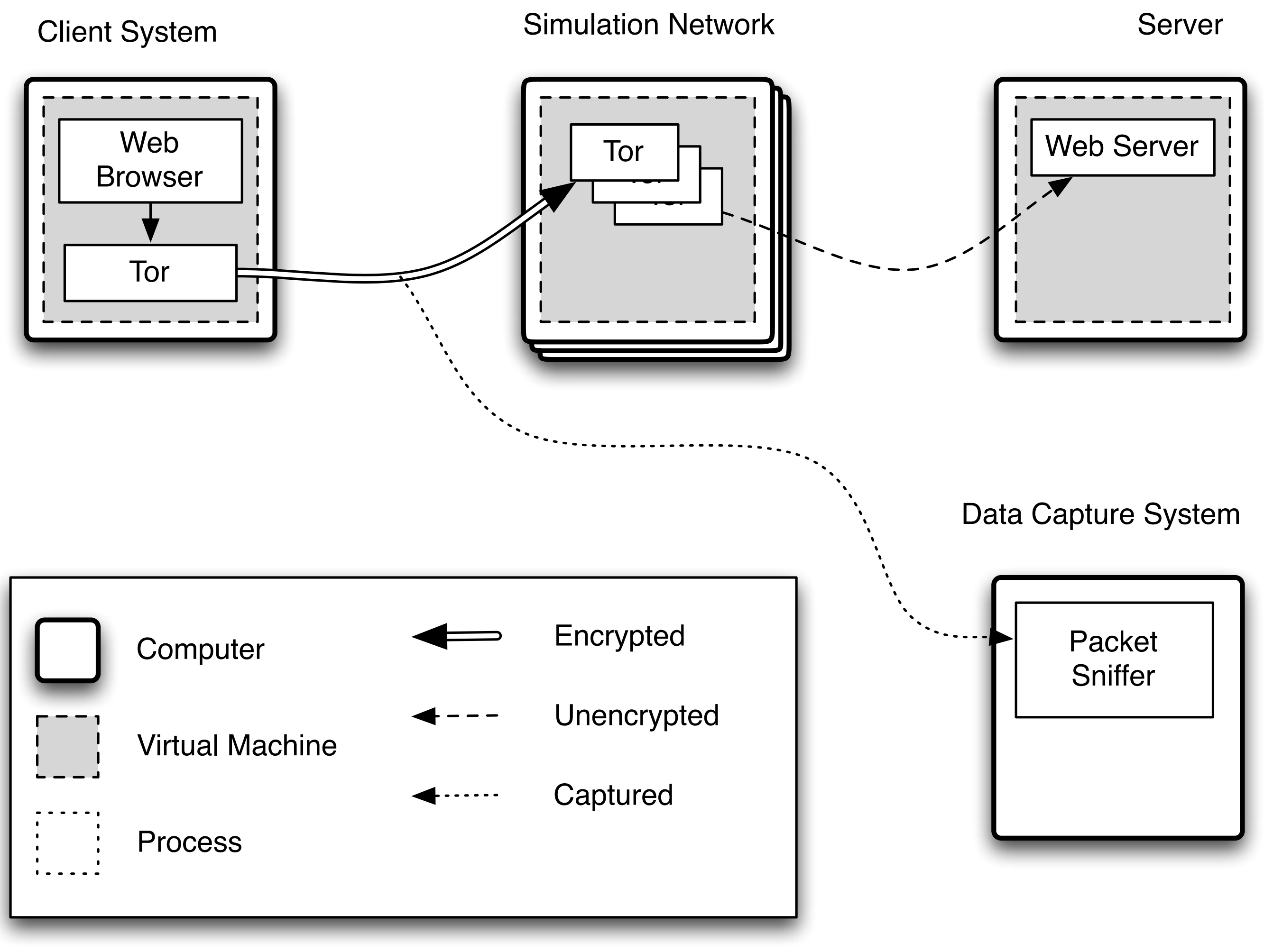

Figure 1.2: Network Diagram

1.4 Systems

A number of systems will be configured both physical and virtual. The *Host Systems*, those running natively on a physical PC with a Virtual Box environment to host a Guest Operating System. The *Guest Systems*, operating systems running inside a Virtual Box environment on top of a Host System, and the Sniffing system, which is responsible for capturing the experiment results.

The name, IP Address and role of each system in the simulation network is shown in Table 1.4.

Host Name	IP Address	Role
Grumman	192.168.0.100	Management and Host of Client System
Arrakis	192.168.0.001	Host of Server System
Richese	192.168.0.102	Host of Tor Network System
Ipyr	192.168.0.200	Sniffing System
Lankiveil	192.168.0.16	Client Guest System
Ginaz	192.168.0.17	Server Guest System
Sikun	192.168.0.18	Tor Network Guest System

1.4.1 Sniffing System

The sniffing system has a very basic role: to capture packets. It is configured with the Ubuntu 10.04 Server Operating system. Most basic features are disabled and tcpdump is configured to run at startup saving all captures to 1mb files in a directory indicating when the system was booted.

1.4.2 Host Systems

All the host systems are configured identically. The following procedure is carried out to prepare a host system.

- Install Ubuntu 10.04 Desktop Operating System
  - Use entire disk
  - Set time zone to GMT +8 WST (Perth)
- Turn off automatic package updates
- Turn off powersaving features
- Install OpenSSH Server
- Add Virtual Box repository to sources.lst
- Update sources and install Virtual Box
- Ensure Virtual Box starts with system and disable all other startup applications

One of the host systems is set aside as a management system, which is responsible for ensuring the experiment runs and restarts it if there are any problems. On this machine a private ssh key is created, and copied to the authorized keys file on all other hosts. This allows the management system to connect to all the other systems and issue commands without user interaction.

The management script runs on the management system to watch the experiment's progress. This is a simple Ruby script which uses SSH to control the other Host systems and opens a socket which waits for a connection that indicates that the experiment has finished. Every time it starts the experiment, it rolls back each virtual machine to a preconfigured snapshot. If the experiment takes too long, say because a virtual machine hangs or something is wrong, it will be started without waiting for the finish condition.

Note that restarting each snapshot ensures that the system clock of each guest operating system is synchronized with that of the host operating system. The Tor network is sensitive to differences in time between hosts and this ensures correct operation.

1.4.3 Guest Systems

Client System

The client system runs Ubuntu 10.04 Desktop Operating System as well as a number of applications that allow it to perform automated web browsing. Tor is configured on this system as well for the Tor part of the experiment but is initially disabled.

The software installed on this system includes:

- Ruby
- Rubygems:
  - Selenium
  - DaemonController
  - RSpec
- Selenium-RC
- Firefox web browser
- Tor

The client operation is controlled by a Ruby script which ensures that the Selenium-RC server is running, executes a number of Selenium browser simulations using RSpec and reports it's success by connecting to an open socket on the management system.

There are 170 simulations which emulate simple website interactions which have different traffic profiles against 30 different websites on the Server system.

For the non-SSL portion of the experiment, Selenium is configured to use a special proxy which automatically accepts self signed certificates. Typical interactions with SSL encrypted websites are signed by a certificate authority, so no user interaction is required to begin browsing.

Server System

The host system runs an apache web server which serves two sets of sample websites, static and dynamic. The static websites are served each from a different numbered directory and accessible over a different virtual host. This means that when a web browser is pointed at the directory `site1.static.torsim` the files in `/var/www/thesis/experiment/simulation/static/site1` are served to the browser.

For the SSL part of the experiment, only the HTTPS port 443 is accessed and all information is sent to the browser using encryption.

The dynamic sample websites change subtly with each attempted access. There are 10 sample websites each a slightly different Ruby Sinatra application.

Tor Simulation

The Tor Simulation Guest is responsible for emulating a Tor network. It runs three Tor directory authorities as well as fifteen relays.

Each Tor instance on the Tor Simulation system is given it's own separate directory for configuration, logging and temporary files.

1.5 Procedure

1.5.1 Preparation

Host Systems

The preparation for the host operating systems is identical except for the hostname and IP address.

Installing the Host operating system:

1. Power on machine

2. Interrupt BIOS boot process
    (a) Load default BIOS settings
    (b) Set boot order to ensure CD-ROM is first
    (c) Disable unnecessary boot devices
    (d) Ensure system clock is correct
3. Boot from Ubuntu 10.04 Desktop cd
4. Select the language you wish to use for installation (English) and click 'Forward'
5. Select the region (Australia/Perth) and click 'Forward'
6. Select an appropriate keymap (USA) and click 'Forward'
7. Select 'Erase and use the entire disk' and click 'Forward'
8. Enter your details, select 'Log in automatically' and click 'Forward'

At this point the installation will begin and should take about 35 minutes. 'Log in automatically' is selected to ensure that the experiment will continue even if power is lost. Once installation is complete remove the CD and reboot the system.

Configuring the Host operating system:

1. Configure the network adapter
2. Install Virtual Box
3. Load the guest operating system virtual machine image

To install Virtual Box, the following procedure is used:

1. Edit the file `/etc/apt/sources.lst` and add the line:

   ```
   deb http://download.virtualbox.org/virtualbox/debian lucid non-free
   ```

2. Install the GPG key for the Virtual Box packages:

```
wget -q http://download.virtualbox.org/virtualbox/debian/oracle_vbox.asc -O- | sudo apt-k
```

3. Then install VirtualBox 3.2 with the command:

   ```
   sudo apt-get update && sudo apt-get install virtualbox-3.2
   ```

For the management system, some additional commands need to be run:

1. Generate a private ssh key for the user with the command: `ssh-keygen`, when prompted for a password enter nothing.
2. SSH to the other host systems on the network, and copy the contents of `$HOME/.ssh/id_rsa.pub` into `$HOME/.ssh/authorized_keys` on the destination system.
3. Install ruby so that the management script can be run:

   ```
   sudo apt-get install ruby rubygems
   ```

4. Ensure the script `run_simulation.rb` is started on login.

Tor System

Install Ubuntu 10.04 Server.

Deploy the experiment files.

Install Tor:

1. Edit the file `/etc/apt/sources.lst` and add the line:

   ```
   deb http://deb.torproject.org/torproject.org lucid main
   ```

2. Install the GPG key for the Tor packages:

```
gpg --keyserver keys.gnupg.net --recv 886DDD89
gpg --export A3C4F0F979CAA22CDBA8F512EE8CBC9E886DDD89 | sudo apt-key add -
```

3. Install Tor:

```
sudo apt-get update && sudo apt-get install tor
```

Client System

Install Ubuntu 10.04 Desktop as in the Host system directions, leaving out the Virtual Box installation.

1. Deploy the experiment files onto the client system.
2. Install Ruby
3. Install the bundler ruby gem with the command:

```
gem install bundler
```

4. Install the required client ruby gems by executing the bundler from the client directory in the deployed experiment files.

```
bundle install
```

5. Ensure that the client script is executed at system startup.

Configuring the Management Script

Once all systems are deployed, the management script needs to be altered to

1.5.2 Execution

Running the experiment is a matter of ensuring all Host systems are powered on, if everything is configured correctly the experiment will continue to run until stopped manually.

2 Appendix

2.1 Manager Script

```ruby
#!/usr/bin/ruby
#
# Experiment manager, makes sure the VMs have their snapshots restored,
# are powered on and then are restarted whenever finished
#
require 'socket'
require 'net/http'

class Manager

  # Set this to true to manage TOR
  USE_TOR = true

  VBOXMANAGE = 'VBoxManage'

  CLIENT_UUID = '638e9eb9-7b5d-471d-b0b1-c528762a9031'
  SERVER_UUID = 'c8c7048b-e860-4e22-973d-a14eaa0d41b2'
  TOR_UUID = 'bf44202b-3b2f-487f-b7d5-968822608e2b'
```

```
CLIENT_HOST = 'grumman'
SERVER_HOST = 'arrakis'
TOR_HOST = 'richese'

CLIENT_ACK_PORT = 65412
SERVER_ACK_PORT = 65413
TOR_ACK_PORT = 65414

# No need to wait for server, client does that for now
FINISH_PORTS = [CLIENT_ACK_PORT]
#FINISH_PORTS = [CLIENT_ACK_PORT, SERVER_ACK_PORT, TOR_ACK_PORT]

HOUR_IN_SECONDS = 3600

# ctor
def initialize
  @tor = USE_TOR
end

#
# Basic operation is, startvms, wait for them to finish, if
# they take too long restart them anyway
#
def run
  while true do
    # Give the experiment 1 hour to continue
    Timeout.timeout(HOUR_IN_SECONDS, Timeout::Error) do
      begin
```

```
          puts 'Starting VMs...'
          restart_vms
          puts 'VMs are running, waiting for experiment to finish...'
          wait_for_experiment
        rescue Timeout::Error
          puts 'Timed out waiting for experiment to finish, restarting forcefully!'
        rescue
          puts "Error during experiment: #{$!.message}"
        end
      end
    end
  end

protected #####################################################################

  #
  # Sets up two sockets to listen on, when both of them
  # have been connected to, consider the experiment over
  #
  def wait_for_experiment
    # Memoize the servers, prevents a socket already listening error
    # if a Timeout exception is thrown above and the sockets aren't
    # closed properly
    @servers ||= FINISH_PORTS.collect do |port|
      puts "Creating listener port #{port}"
      TCPServer.new(port)
    end
    servers = @servers.dup
    begin
      until servers.empty?
        servers.delete_if do |server|
          socket = server.accept_nonblock
          puts "Got socket #{socket}"
          socket.close
          true
        end
      end
    rescue Errno::EAGAIN, Errno::EWOULDBLOCK, Errno::ECONNABORTED, Errno::EPROTO, Errno::EIN
      IO.select(servers)
      retry
    end
    puts "All clients finished"
    true
  end

  def remote_command(host, command)
    system "ssh john@#{host} '#{command}'"
  end

  def restart_vms
    # Shutdown
    puts 'Shutting down! .............'
    puts '-- client --'
    remote_command(CLIENT_HOST, shutdownvm(CLIENT_UUID))
    puts '-- server --'
    remote_command(SERVER_HOST, shutdownvm(SERVER_UUID))
```

```
    if tor?
      puts '-- tor --'
      remote_command(TOR_HOST, shutdownvm(TOR_UUID))
    end

    # Restore
    puts 'Restoring from snapshot! ..............'
    puts '-- client --'
    remote_command(CLIENT_HOST, restorevm(CLIENT_UUID))
    puts '-- server --'
    remote_command(SERVER_HOST, restorevm(SERVER_UUID))
    if tor?
      puts '-- tor --'
      remote_command(TOR_HOST, restorevm(TOR_UUID))
    end

    # Power up!
    puts 'Powering on! ..............'
    puts '-- client --'
    remote_command(CLIENT_HOST, startvm(CLIENT_UUID))
    puts '-- server --'
    remote_command(SERVER_HOST, startvm(SERVER_UUID))
    if tor?
      puts '-- tor --'
      remote_command(TOR_HOST, startvm(TOR_UUID))
    end
  end
  # Make the command to restore a vm snapshot
  def restorevm(uuid)
    "#{VBOXMANAGE} snapshot #{uuid} restore experiment"
  end

  # Make the command to start a virtual machine
  def startvm(uuid)
    "#{VBOXMANAGE} startvm --type gui #{uuid}"
  end

  def shutdownvm(uuid)
    "#{VBOXMANAGE} controlvm #{uuid} poweroff"
  end

  # Does this experiment use tor
  def tor?
    @tor
  end

end

Manager.new.run
```

2.2 Client Script

```
#!/usr/bin/ruby
#
# Ruby client simulation script, ensures selenium-rc is running and launches
# spec against scripts/
#

require 'rubygems'
require 'daemon_controller'
require 'socket'
require 'net/http'

class TorSimulationClient

  # What is the name of the server
  SERVER_HOST='ginaz'

  # We are patient...
  TIMEOUT = 500
  PING_INTERVAL = 3

  # Where is selenium located
  SELENIUM_HOST = 'localhost'
  SELENIUM_PORT = 4444

  # Commands to control selenium
  SHUTDOWN_SELENIUM_COMMAND = 'shutDownSeleniumServer'
  PING_SELENIUM = \
    lambda { TCPSocket.new(SELENIUM_HOST, SELENIUM_PORT) }
  SHUTDOWN_SELENIUM = \
    "wget -t 30 -O /dev/null http://#{SELENIUM_HOST}:#{SELENIUM_PORT}/selenium-server/driver
  # DaemonController doesn't take a proc for this argument :(
    #lambda { Net::HTTP.get(SELENIUM_HOST, "/selenium-server/driver/?cmd=#{SHUTDOWN_SELENIU

  def run
    # Start the RC server
    @selenium_rc = DaemonController.new(
      :identifier => 'Selenium RC Server',
      :start_command => 'sh ./selenium-rc.sh',
      :stop_command => SHUTDOWN_SELENIUM,
      :ping_command => PING_SELENIUM,
      :pid_file => 'selenium.pid',
      :log_file => 'selenium.log',
      :start_timeout => TIMEOUT,
      :stop_timeout => TIMEOUT,
      :log_file_activity_timeout => TIMEOUT,
      :ping_interval => PING_INTERVAL)

    puts 'Starting selenium-rc...'
    @selenium_rc.start

    puts 'Waiting for server...'
    wait_for_server

    puts 'Beginning simulations...'
    system 'spec -c -f nested scripts/*_spec.rb'

    puts "Registering with manager that we're finished..."
    TCPSocket.new('grumman', 65412)
```

```
rescue
  puts "Error #{$!.message}"
ensure
  @selenium_rc.stop
  # These get left around some time, remove them
  File.unlink 'selenium.pid.lock'
end

# Wait for the server to start, by connecting to apache and seeing if it's
# ready
def wait_for_server
  Timeout.timeout(TIMEOUT, Timeout::Error) do
    stop = nil
    until stop do
      begin
        Net::HTTP.get(SERVER_HOST, '/')
        puts "Server #{SERVER_HOST} is alive!"
        stop = true

      rescue Timeout::Error
        puts 'Timed out waiting for server, continuing anyway...'
        stop = true

      rescue
        puts "Error #{$!.message}. Waiting for server..."
        sleep(PING_INTERVAL)
      end
    end
    end
  end
end

# Always run
TorSimulationClient.new.run
```

2.3 Selenium-RC Script

```
#!/bin/sh
#
# Launch Selenium-RC using nohup and save the pid to a file, used
# by DaemonController in client.rb
#
# -trustAllSSLCertificates Use the *firefoxproxy to automatically trust SSL certs
# -singleWindow Use a single firefox window
# -timeout Generous timeout
#
SELENIUM_SERVER="selenium-remote-control-1.0.3/selenium-server-1.0.3/selenium-server.jar"
#SELENIUM_SEVER="selenium-server-standalone-2.0a5.jar"

nohup java -jar "$SELENIUM_SERVER" -trustAllSSLCertificates -singleWindow -timeout 300 -Dsoc
echo $! > selenium.pid
```

2.4 Selenium-RC Script (with Tor)

```
#!/bin/sh

#

# Launch Selenium-RC using nohup and save the pid to a file, used

# by DaemonController in client.rb

#

# -trustAllSSLCertificates Use the *firefoxproxy to automatically trust SSL certs

# -singleWindow Use a single firefox window

# -timeout Generous timeout

#

SELENIUM_SERVER="selenium-remote-control-1.0.3/selenium-server-1.0.3/selenium-server.jar"

#SELENIUM_SEVER="selenium-server-standalone-2.0a5.jar"


nohup java -jar "$SELENIUM_SERVER" -trustAllSSLCertificates -singleWindow -timeout 300 </dev

echo $! > selenium.pid
```

2.5 Hosts file

```
## Static hosts


# Laptop

192.168.0.15 ix


# Experiment network (host operating systems)

192.168.0.100 grumman

192.168.0.101 arrakis

192.168.0.102 richese
```

```
192.168.0.103 romo
192.168.0.200 ipyr

# Experiment network (guest operating systems)
192.168.0.13 ecaz
192.168.0.16 lankiveil
192.168.0.17 ginaz
192.168.0.18 sikun
192.168.0.19 palma

# Tor simulation webservers
192.168.0.17  site1.static.torsim
192.168.0.17  site2.static.torsim
192.168.0.17  site3.static.torsim
192.168.0.17  site4.static.torsim
192.168.0.17  site5.static.torsim
192.168.0.17  site6.static.torsim
192.168.0.17  site7.static.torsim
192.168.0.17  site8.static.torsim
192.168.0.17  site9.static.torsim
192.168.0.17  site10.static.torsim
192.168.0.17  site11.static.torsim
192.168.0.17  site12.static.torsim
192.168.0.17  site13.static.torsim
192.168.0.17  site14.static.torsim
192.168.0.17  site15.static.torsim
192.168.0.17  site16.static.torsim
192.168.0.17  site17.static.torsim
```

```
192.168.0.17  site18.static.torsim
192.168.0.17  site19.static.torsim
192.168.0.17  site20.static.torsim
192.168.0.17  site21.static.torsim
192.168.0.17  site22.static.torsim
192.168.0.17  site22.static.torsim
192.168.0.17  site23.static.torsim
192.168.0.17  site24.static.torsim
192.168.0.17  site25.static.torsim
192.168.0.17  site26.static.torsim
192.168.0.17  site27.static.torsim
192.168.0.17  site28.static.torsim
192.168.0.17  site29.static.torsim
192.168.0.17  site30.static.torsim

# Dynamic simulation component (on different ports)
192.168.0.17  site50.dynamic.torsim
192.168.0.17  site51.dynamic.torsim
192.168.0.17  site52.dynamic.torsim
192.168.0.17  site52.dynamic.torsim
192.168.0.17  site53.dynamic.torsim
192.168.0.17  site54.dynamic.torsim
192.168.0.17  site55.dynamic.torsim
192.168.0.17  site56.dynamic.torsim
192.168.0.17  site57.dynamic.torsim
192.168.0.17  site58.dynamic.torsim
192.168.0.17  site59.dynamic.torsim
```

```
<IfModule mod_ssl.c>

# Basic static sites, we can use virtual document root for these!
<VirtualHost *:80>
        # Virtual setup
        UseCanonicalName Off
        VirtualDocumentRoot /home/etw/thesis/experiment/simulation/site/%-2/%-3/
</VirtualHost>

<VirtualHost *:443>
        # Turn SSL on with snake oil certs
        SSLEngine on
        SSLCertificateFile    /etc/ssl/certs/ssl-cert-snakeoil.pem
        SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key

        # Virtual setup
        UseCanonicalName Off
        VirtualDocumentRoot /home/etw/thesis/experiment/simulation/site/%-2/%-3/
</VirtualHost>

# Dynamic sites, since these use passenger/sinatra we have to specify the document root for
<VirtualHost *:443>
        # Turn SSL on with snake oil certs
        SSLEngine on
        SSLCertificateFile    /etc/ssl/certs/ssl-cert-snakeoil.pem
        SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
        ServerName site50.dynamic.torsim
        DocumentRoot /home/etw/thesis/experiment/simulation/site/dynamic/site50/public/
</VirtualHost>
<VirtualHost *:443>
        # Turn SSL on with snake oil certs
        SSLEngine on
        SSLCertificateFile    /etc/ssl/certs/ssl-cert-snakeoil.pem
        SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
        ServerName site51.dynamic.torsim
        DocumentRoot /home/etw/thesis/experiment/simulation/site/dynamic/site51/public/
</VirtualHost>
<VirtualHost *:443>
        # Turn SSL on with snake oil certs
        SSLEngine on
        SSLCertificateFile    /etc/ssl/certs/ssl-cert-snakeoil.pem
        SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
        ServerName site52.dynamic.torsim
        DocumentRoot /home/etw/thesis/experiment/simulation/site/dynamic/site52/public/
</VirtualHost>
<VirtualHost *:443>
        # Turn SSL on with snake oil certs
        SSLEngine on
        SSLCertificateFile    /etc/ssl/certs/ssl-cert-snakeoil.pem
        SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
        ServerName site53.dynamic.torsim
        DocumentRoot /home/etw/thesis/experiment/simulation/site/dynamic/site53/public/
</VirtualHost>
<VirtualHost *:443>
```

```
        # Turn SSL on with snake oil certs
        SSLEngine on
        SSLCertificateFile    /etc/ssl/certs/ssl-cert-snakeoil.pem
        SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
        ServerName site54.dynamic.torsim
        DocumentRoot /home/etw/thesis/experiment/simulation/site/dynamic/site54/public/
</VirtualHost>
<VirtualHost *:443>
        # Turn SSL on with snake oil certs
        SSLEngine on
        SSLCertificateFile    /etc/ssl/certs/ssl-cert-snakeoil.pem
        SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
        ServerName site55.dynamic.torsim
        DocumentRoot /home/etw/thesis/experiment/simulation/site/dynamic/site55/public/
</VirtualHost>
<VirtualHost *:443>
        # Turn SSL on with snake oil certs
        SSLEngine on
        SSLCertificateFile    /etc/ssl/certs/ssl-cert-snakeoil.pem
        SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
        ServerName site56.dynamic.torsim
        DocumentRoot /home/etw/thesis/experiment/simulation/site/dynamic/site56/public/
</VirtualHost>
<VirtualHost *:443>
        # Turn SSL on with snake oil certs
        SSLEngine on
        SSLCertificateFile    /etc/ssl/certs/ssl-cert-snakeoil.pem
        SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
        ServerName site57.dynamic.torsim
        DocumentRoot /home/etw/thesis/experiment/simulation/site/dynamic/site57/public/
</VirtualHost>
<VirtualHost *:443>
        # Turn SSL on with snake oil certs
        SSLEngine on
        SSLCertificateFile    /etc/ssl/certs/ssl-cert-snakeoil.pem
        SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
        ServerName site58.dynamic.torsim
        DocumentRoot /home/etw/thesis/experiment/simulation/site/dynamic/site58/public/
</VirtualHost>
<VirtualHost *:443>
        # Turn SSL on with snake oil certs
        SSLEngine on
        SSLCertificateFile    /etc/ssl/certs/ssl-cert-snakeoil.pem
        SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
        ServerName site59.dynamic.torsim
        DocumentRoot /home/etw/thesis/experiment/simulation/site/dynamic/site59/public/
</VirtualHost>

<VirtualHost *:80>
        ServerName site50.dynamic.torsim
        DocumentRoot /home/etw/thesis/experiment/simulation/site/dynamic/site50/public/
</VirtualHost>
<VirtualHost *:80>
        ServerName site51.dynamic.torsim
        DocumentRoot /home/etw/thesis/experiment/simulation/site/dynamic/site51/public/
</VirtualHost>
```

```
<VirtualHost *:80>
        ServerName site52.dynamic.torsim
        DocumentRoot /home/etw/thesis/experiment/simulation/site/dynamic/site52/public/
</VirtualHost>
<VirtualHost *:80>
        ServerName site53.dynamic.torsim
        DocumentRoot /home/etw/thesis/experiment/simulation/site/dynamic/site53/public/
</VirtualHost>
<VirtualHost *:80>
        ServerName site54.dynamic.torsim
        DocumentRoot /home/etw/thesis/experiment/simulation/site/dynamic/site54/public/
</VirtualHost>
<VirtualHost *:80>
        ServerName site55.dynamic.torsim
        DocumentRoot /home/etw/thesis/experiment/simulation/site/dynamic/site55/public/
</VirtualHost>
<VirtualHost *:80>
        ServerName site56.dynamic.torsim
        DocumentRoot /home/etw/thesis/experiment/simulation/site/dynamic/site56/public/
</VirtualHost>
<VirtualHost *:80>
        ServerName site57.dynamic.torsim
        DocumentRoot /home/etw/thesis/experiment/simulation/site/dynamic/site57/public/
</VirtualHost>
<VirtualHost *:80>
        ServerName site58.dynamic.torsim
        DocumentRoot /home/etw/thesis/experiment/simulation/site/dynamic/site58/public/
</VirtualHost>
```

```
<VirtualHost *:80>
        ServerName site59.dynamic.torsim
        DocumentRoot /home/etw/thesis/experiment/simulation/site/dynamic/site59/public/
</VirtualHost>

</IfModule>
```

2.7 Tor Setup Script

```
#!/bin/bash

#set -e

## Trivial script to set up a private Tor network.

## This many authorities/relays/clients will be started
AUTHORITIES=3
RELAYS=15
CLIENTS=1

## Find the executables here
TOR=/usr/sbin/tor
GENCERT=/usr/bin/tor-gencert

## If you want to run with a lowered ulimit -n to make sure
## you're not accidentally wasting your resources, set the
## ConnLimit config value here so that Tor still starts. This
## must be substantially higher than the number auf authorities
## + relays.
CONNLIMIT=90

## Address to use for systems in the tor network
ADDRESS=192.168.0.18

################################################################

WD=`pwd`/work

mkdir -p $WD/authorities/
mkdir -p $WD/relays/
mkdir -p $WD/clients/

# Set up authorities

DIRSERVER_LINE=

# Make authority keys
NUM=$AUTHORITIES
while [ $NUM -gt 0 ]; do
        path=$WD/authorities/auth$NUM;
        mkdir -p $path;
        cd $path;
        let ORPORT=3000+$NUM
        let DIRPORT=4000+$NUM
        cat <<-EOF >$path/torrc.tmp
        DirServer test $ADDRESS:1 0000000000000000000000000000000000000000
        OrPort 1
        EOF

        FP=`$TOR --quiet --list-fingerprint --DataDirectory $path -f torrc.tmp \
        | cut -f 2,3,4,5,6,7,8,9,10,11 -d " " | sed 's/ //g'`;

        rm -rf $path/torrc.tmp

        # Make a dummy password for this authority
        echo $NUM$NUM$NUM$NUM > password

        exec 5<> password
        $GENCERT --create-identity-key --passphrase-fd 5
        exec 5>&-

        mv authority_certificate authority_signing_key keys/

        V3ID=`grep fingerprint keys/authority_certificate | cut -f 2 -d " "`;

        DIRSERVER_LINE="DirServer authority$NUM v3ident=$V3ID orport=$ORPORT \
        no-v2 $ADDRESS:$DIRPORT $FP"$'\n'"$DIRSERVER_LINE";

        let NUM=$NUM-1
done

# Configure them

NUM=$AUTHORITIES
```

```
while [ $NUM -gt 0 ]; do

        path=$WD/authorities/auth$NUM;
        let ORPORT=3000+$NUM
        let DIRPORT=4000+$NUM

        # Make the config file
        cat <<-EOF >$path/torrc
        TestingTorNetwork 1
        DataDirectory $path
        Log notice file $path/notice.log
        Nickname authority$NUM
        RunAsDaemon 1
        SocksPort 0
        OrPort $ORPORT
        Address $ADDRESS
        DirPort $DIRPORT
        ConnLimit $CONNLIMIT
        AuthoritativeDirectory 1
        V3AuthoritativeDirectory 1
        ContactInfo auth$NUM@test.test
        ExitPolicy reject *:*
        $DIRSERVER_LINE
        EOF

        let NUM=$NUM-1
done
```

```
# Set up relays

NUM=$RELAYS
while [ $NUM -gt 0 ]; do

        path=$WD/relays/relay$NUM;
        mkdir -p $path;
        cd $path
        let ORPORT=5000+$NUM
        let DIRPORT=6000+$NUM

        # Make the config file
        cat <<-EOF >$path/torrc
        TestingTorNetwork 1
        DataDirectory $path
        Log notice file $path/notice.log
        Nickname relay$NUM
        RunAsDaemon 1
        SocksPort 0
        OrPort $ORPORT
        ConnLimit $CONNLIMIT
        Address $ADDRESS
        DirPort $DIRPORT
        $DIRSERVER_LINE
        EOF

        let NUM=$NUM-1
done
```

```
# Set up clients

NUM=$CLIENTS
while [ $NUM -gt 0 ]; do

        path=$WD/clients/client$NUM;
        mkdir -p $path;
        cd $path

        let SOCKSPORT=10000+$NUM

        # Make the config file
        cat <<-EOF >$path/torrc
        TestingTorNetwork 1
        DataDirectory $path
        RunAsDaemon 1
        ConnLimit $CONNLIMIT
        Log notice file $path/notice.log
        SocksPort $SOCKSPORT
        $DIRSERVER_LINE
        EOF

        let NUM=$NUM-1
done
```

2.8 Tor Running Script

```bash
#!/bin/bash


## Run all the tor servers set up in ./work setup by make_private_network.sh


## Find the executables here

TOR=/usr/sbin/tor


###################################################################


WD=`pwd`/work


cd $WD


find . -name torrc -exec bash -c "$TOR -f $WD/{} &" \;
```

REFERENCES

Lewman, A., & Harris, M. D. (yearmonthday). TheOnionRouter/TorFAQ - tor bug tracker. Retrieved May, 2010 from `https://trac.torproject.org/projects/tor/wiki/TheOnionRouter/TorFAQ#HowdoIsetupmyownprivateTornetwork`.