

Crea tu propia red privada de TOR – Emulación de TOR con Chutney

noviembre 20, 2014 [adastra](#) [Deja un comentario](#) [Go to comments](#)

Somos muchos los que nos interesa saber cómo funciona TOR, pero también somos muchos los que sabemos que existen varios riesgos cuando entramos a ese tipo de redes cuando lo único que queremos es hacer pruebas, por ese motivo, una buena forma de comprender el funcionamiento de todos los elementos que conforman TOR, sin acceder directamente a la red, es por medio de herramientas y librerías que permiten simular y/o emular la red.

En primer lugar, es necesario entender la diferencia entre simular y emular, ambos términos a veces suelen utilizarse como si se tratase de sinónimos, pero no son lo mismo. Cuando simulamos un sistema, contamos con **modelo** muy específico que aplica a un contexto particular, por ejemplo, podemos simular el rendimiento midiendo los tiempos de respuesta y haciendo estimaciones sobre su comportamiento con una mayor carga o con más o menos recursos, es decir, una simulación es un mecanismo que permite modelar un sistema en unas condiciones determinadas y es muy útil para hacer estimaciones. Por otro lado, cuando se habla de emular, se **duplica** el sistema y se ejecuta de forma independiente al sistema original, es decir, se produce una replica del sistema y se puede observar y analizar su comportamiento, dando resultados mucho más exactos que los que puede proporcionar una simulación.

Ahora bien, las simulaciones y emulaciones son términos que seguramente resultarán bastante conocidos y utilizados a los administradores de sistemas y redes, ya que existen varias herramientas que emplean ambos mecanismos para medir los límites de un sistema o el número de clientes concurrentes que pueden estar conectados a una red sin deteriorar su rendimiento. En este sentido, es posible utilizar algunas de estas mismas herramientas para probar el funcionamiento de TOR sin necesidad de conectarse directamente a las autoridades de directorio oficiales. El objetivo de este artículo y el siguiente, es enseñar el uso de dos herramientas que son recomendadas por el equipo de TOR para hacer pruebas sobre la red, dichas herramientas son Chutney y Shadow, la primera es una herramienta de emulación y la segunda de simulación. En esta entrada nos centraremos en Chutney y en la próxima se hablará sobre Shadow.

Instalación y uso de Chutney

Chutney es una herramienta que permite emular y configurar una red privada con TOR muy rápidamente, permitiendo crear varios escenarios en los que Chutney es capaz de levantar autoridades de directorio, repetidores, clientes, bridges y cualquier elemento adicional que conforma la red de TOR. Usar Chutney no es complicado, de hecho es una herramienta bastante sencilla e intuitiva, pero para configurar escenarios de prueba, es necesario conocer bastante bien las propiedades de configuración que admite TOR, es decir, aquellas que típicamente se incluyen en el fichero de configuración “torrc”. El proyecto se encuentra ubicado en la siguiente ruta: <https://gitweb.torproject.org/chutney.git> y para comenzar a usarlo, basta con clonar el repositorio GIT y lanzar la utilidad “chutney”

```
>git clone https://git.torproject.org/chutney.git
Clonar en «chutney»...
remote: Counting objects: 364, done.
remote: Compressing objects: 100% (170/170), done.
remote: Total 364 (delta 180), reused 282 (delta 137)
Receiving objects: 100% (364/364), 58.68 KiB | 0 bytes/s, done.
Resolving deltas: 100% (180/180), done.
Checking connectivity... hecho.
>cd chutney
>./chutney
Error: Not enough arguments given.
Usage: chutney {command} {networkfile}
Known commands are: configure hup restart start status stop verify
```

Para ejecutar “chutney” se debe indicar un comando y un fichero de configuración de red. Los comandos disponibles son “configure”, “hup”, “restart”, “start”, “status”, “stop” y “verify”. Además, se debe indicar la ruta del fichero de configuración de red como segundo argumento del comando.

Para que la herramienta funcione correctamente, el comando “tor” debe ser un comando valido para el usuario que lo ejecuta, es decir, que TOR se debe encontrar instalado en el sistema y el usuario en cuestión debe de tener los privilegios suficientes para poder ejecutarlo, de lo contrario, “chutney” enseñará el siguiente mensaje.

```
>./chutney configure networks/basic
Starting nodes
Cannot find tor binary ‘tor’. Use CHUTNEY_TOR environment variable to set the path, or put the
binary into $PATH.
```

Basta con establecer la ruta donde se encuentra el ejecutable en la variable de entorno PATH.

```
>nano /home/adastra/.bashrc
export PATH="$PATH:/TOR/tor-browser_40/Browser/TorBrowser/Tor/tor"
>source ~/.bashrc
```

Si TOR se ha instalado desde algún repositorio de Debian, CentOS o Fedora, no debería haber mayores problemas, sin embargo se recomienda instalar la última versión de TOR disponible, preferiblemente desde el código fuente.

Después de tener todos los elementos instalados y listos para ser utilizados, el primer paso en la puesta en marcha de “chutney” es invocar al comando “configure” para crear todos los directorios y las claves necesarias para instanciar las autoridades de directorio en la máquina donde se ejecuta la herramienta. Dicho comando se debe ejecutar sobre un directorio de configuración de red y en este caso, “chutney” cuenta con algunos ficheros de ejemplo que pueden ser útiles para probar la herramienta. Dichos ficheros se encuentran incluidos en el directorio “networks”.

Cuando se ejecuta el comando “configure”, la herramienta se encarga de crear automáticamente el directorio “net”, el cual incluye todos los elementos necesarios para emular la red de TOR.

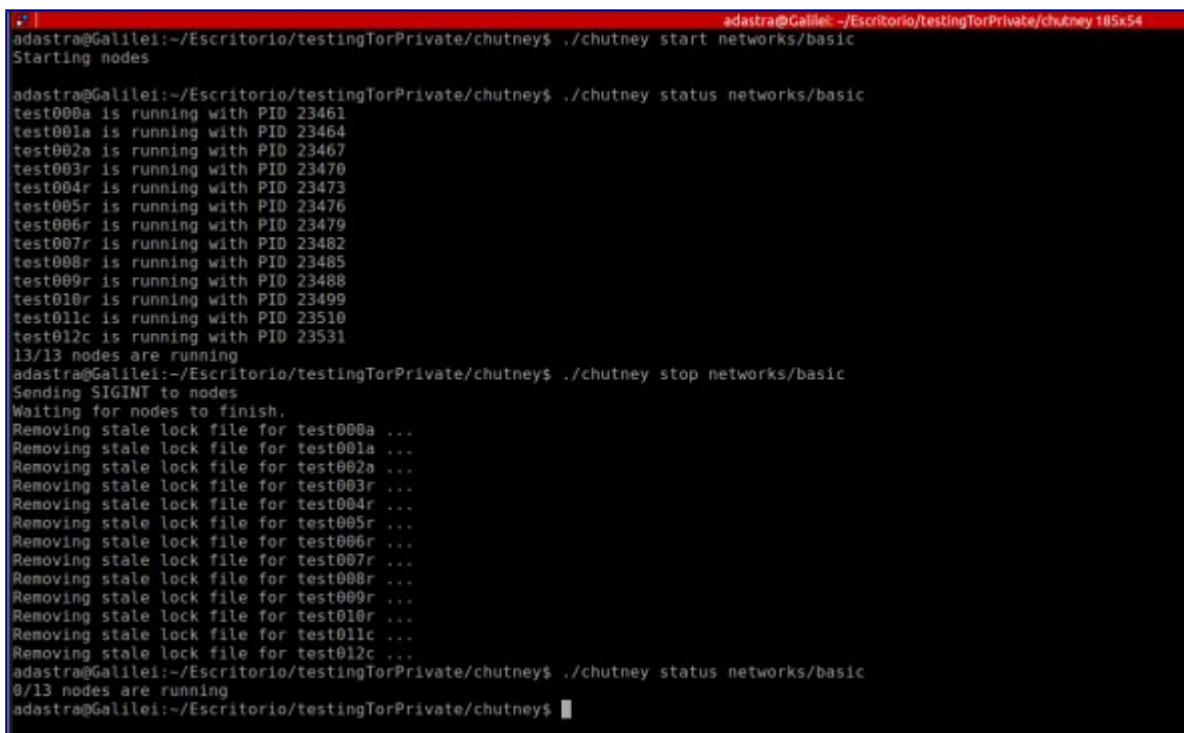
```
>./chutney configure networks/basic
```

```

Creating identity key /chutney/net/nodes/000a/keys/authority_identity_key for test000a with tor-
gencert --create-identity-key --passphrase-fd 0 -i /chutney/net/nodes/000a/keys/authority_identity_key
-s /chutney/net/nodes/000a/keys/authority_signing_key -c
/chutney/net/nodes/000a/keys/authority_certificate -m 12 -a 127.0.0.1:7000
Creating identity key
/chutney/net/nodes/001a/keys/authority_identity_key for test001a with tor-gencert --create-identity-
key --passphrase-fd 0 -i /chutney/net/nodes/001a/keys/authority_identity_key -s
/chutney/net/nodes/001a/keys/authority_signing_key -c
/chutney/net/nodes/001a/keys/authority_certificate -m 12 -a 127.0.0.1:7001
Creating identity key
/chutney/net/nodes/002a/keys/authority_identity_key for test002a with tor-gencert --create-identity-
key --passphrase-fd 0 -i /chutney/net/nodes/002a/keys/authority_identity_key -s
/chutney/net/nodes/002a/keys/authority_signing_key -c
/chutney/net/nodes/002a/keys/authority_certificate -m 12 -a 127.0.0.1:7002
The tor binary at 'tor' does not support the option in the torrc line: 'TestingDirAuthVoteExit *'
The tor binary at 'tor' does not support the option in the torrc line:
'TestingDirAuthVoteExit *'
The tor binary at 'tor' does not support the option in the torrc line:
'TestingDirAuthVoteExit *'

```

Como se puede apreciar en las últimas líneas, la opción de configuración “TestingDirAuthVoteExit” no se ha encontrado y aunque es posible seguir trabajando con “chutney” sin problemas a pesar de este error, se trata de una propiedad de configuración que se ha incluido a partir de la versión “2.6-alpha” de TOR y a la fecha de redactar este artículo, aun no se encuentra disponible en la versión estable. Ahora que se encuentra la red privada configurada, se puede controlar con los comandos “start”, “stop” y “status” tal como enseña la siguiente imagen.



```

adastra@Galilei: ~/Escritorio/testingTorPrivate/chutney 185x54
adastra@Galilei:~/Escritorio/testingTorPrivate/chutney$ ./chutney start networks/basic
Starting nodes

adastra@Galilei:~/Escritorio/testingTorPrivate/chutney$ ./chutney status networks/basic
test000a is running with PID 23461
test001a is running with PID 23464
test002a is running with PID 23467
test003r is running with PID 23470
test004r is running with PID 23473
test005r is running with PID 23476
test006r is running with PID 23479
test007r is running with PID 23482
test008r is running with PID 23485
test009r is running with PID 23488
test010r is running with PID 23499
test011c is running with PID 23510
test012c is running with PID 23531
13/13 nodes are running
adastra@Galilei:~/Escritorio/testingTorPrivate/chutney$ ./chutney stop networks/basic
Sending SIGINT to nodes
Waiting for nodes to finish.
Removing stale lock file for test000a ...
Removing stale lock file for test001a ...
Removing stale lock file for test002a ...
Removing stale lock file for test003r ...
Removing stale lock file for test004r ...
Removing stale lock file for test005r ...
Removing stale lock file for test006r ...
Removing stale lock file for test007r ...
Removing stale lock file for test008r ...
Removing stale lock file for test009r ...
Removing stale lock file for test010r ...
Removing stale lock file for test011c ...
Removing stale lock file for test012c ...
adastra@Galilei:~/Escritorio/testingTorPrivate/chutney$ ./chutney status networks/basic
0/13 nodes are running
adastra@Galilei:~/Escritorio/testingTorPrivate/chutney$

```

Comandos disponibles en chutney

La configuración básica funciona correctamente y como se puede ver, se han creado una serie de procesos que representan instancias de TOR que se están ejecutando en la máquina local. No obstante, no se trata de instancias de TOR que se levantan con las propiedades por defecto de TOR Browser, se trata de instancias especialmente configuradas para levantar clientes, autoridades de directorio, repetidores de salida, bridges, etc. En este caso concreto, el contenido del fichero “networks/basic” tiene lo siguiente.

```
Authority = Node(tag="a", authority=1, relay=1, torrc="authority.tmpl")
Relay = Node(tag="r", relay=1, torrc="relay.tmpl")
Client = Node(tag="c", torrc="client.tmpl")
NODES = Authority.getN(3) + Relay.getN(8) + Client.getN(2)
ConfigureNodes(NODES)
```

Aunque pueda parecer complejo, esta es la estructura de cualquier fichero de configuración de red de la herramienta y en este caso concreto, se están creando 3 instancias de TOR que actuarán como autoridades de directorio, 8 como repetidores y 2 como clientes, lo que nos da un total de 13 nodos ejecutándose en la máquina local.

La estructura del fichero utiliza la API de “chutney” que se encuentra escrita en Python y cuya clase principal es “Node”, en la que se permite crear una instancia de TOR con características muy concretas y además, cada una de dichas instancias contará con su propio fichero de configuración de TOR (torrc). Si se mira con atención el fichero de configuración anterior, se puede apreciar que se compone de una serie de instancias de la clase “Node” las cuales son “clonadas” utilizando un patrón “singleton” con el método “getN” en el que se indica el número de instancias que se debe crear. Finalmente, la utilidad “ConfigureNodes” se encarga de configurar todas y cada una de las referencias declaradas.

Configuración personalizada de Chutney

Hasta este punto se ha podido apreciar que configurar “chutney” no es demasiado complejo, sin embargo se ha utilizado una configuración que viene por defecto en el fichero “networks/basic”. Para hacer pruebas un poco más robustas y que posteriormente puedan servir para detectar errores en el funcionamiento de TOR, es necesario aprender a crear ficheros de configuración de red personalizados, algo que tampoco es demasiado complejo si se conocen las principales propiedades que se pueden utilizar en TOR.

En primer lugar, es importante anotar que el constructor de la clase “Node” recibe dos argumentos que identifican la configuración que se debe aplicar en la instancia de TOR, el primero es “tag”, el cual indica si se trata de un repetidor, un cliente, una autoridad de directorio o un bridge y el segundo es el parámetro “torrc”, el cual recibe un nombre de fichero con extensión “tmpl”, el cual representa una plantilla con las propiedades soportadas por TOR.

En el ejemplo anterior, el fichero “networks/basic” ha utilizado las plantillas “authority.tmpl”, “relay.tmpl” y “client.tmpl” y dichas plantillas se encuentran definidas en el directorio “<CHUTNEY_DIR>/templates/”. Para que el lector se haga una idea sobre lo que pueden incluir dichos ficheros, el siguiente es el contenido de la plantilla “authority.tmpl”

```
${include:relay.tmpl}
```

```
AuthoritativeDirectory 1
V3AuthoritativeDirectory 1
ContactInfo auth${nodenum}@test.test
ExitPolicy reject *:~
TestingV3AuthInitialVotingInterval 300
TestingV3AuthInitialVoteDelay 2
TestingV3AuthInitialDistDelay 2
TestingV3AuthVotingStartOffset 0
# Work around situations where the Exit and Guard flags aren't being set
# These flags are set eventually, but it takes ~30 minutes
# We could be more precise here, but it's easiest just to vote everything
# Clients are sensible enough to filter out Exits without any exit ports,
# and Guards without ORPorts
# If your tor doesn't recognise TestingDirAuthVoteExit, update your chutney
# to a version that includes the issue-13161-check-torrc-options features
TestingDirAuthVoteExit *
TestingDirAuthVoteGuard *
```

Efectivamente, se trata de ficheros muy simples que siguen la misma sintaxis y estructura que el fichero de configuración “torrc”. La única diferencia es que los ficheros plantilla de “chutney” deben incluir una declaración al principio del fichero en el caso de que sea necesario incluir otras plantillas y además, es posible acceder a variables globales “chutney” con “\${variable}” tal como se ha visto en el fichero.

Para probar cualquier tipo de configuración es necesario conocer muy bien las propiedades de configuración de TOR y jugar con ellas!

Chutney ya se encarga de crear el entorno por nosotros y emular la red de TOR en nuestro ordenador, con tantas instancias como queramos. Además, como se verá en un próximo artículo, también es posible utilizar Chutney en otras máquinas del segmento de red para tener una emulación mucho más aproximada del comportamiento de TOR, desde un entorno controlado y sin necesidad de estar conectado a Internet.

Herramientas como Chutney son muy utilizadas actualmente por hackers e investigadores con el fin de estudiar el comportamiento de TOR en un entorno aislado para que de esta forma, sea más sencillo encontrar fallos en su implementación sin afectar directamente a la red. Es una herramienta que desde luego es muy recomendada para aquellas personas que quieran probar diferentes configuraciones de TOR en su red local.

Saludos y Happy Hack!