IUA – Instituto Universitario Aeronáutico – Facultad de Ingeniería

Trabajo Final de Grado

"Redes anónimas como instrumento para la conformación de un ciber-ataque sobre una infraestructura crítica: Sistema de Radarización de Tráfico Aéreo"

Maximiliano Delfino

DEDICATORIA

A mi madre Graciela

Por haberme apoyado en los estudios desde pequeño, aconsejado y enseñado tantas cosas durante el tiempo que pude compartir con ella.

A mi padre Orlando

Por seguir apoyándome en todo lo que me propongo y sobre todo después de momentos tan difíciles.

A toda mi familia

Por estar siempre para lo que necesité relacionado a los estudios.

A mis compañeros y amigos

Por haber compartido 5 años de cursado, de discusiones, de aprendizaje y de amistad.

AGRADECIMIENTOS

Este trabajo de investigación se desarrolló bajo la tutoría del Ingeniero Eduardo Casanovas, a quién me gustaría expresar mi agradecimiento por hacer posible este estudio, por la paciencia, tiempo y dedicación a lo largo de todo el proyecto.

A mis profesores, que compartieron sus conocimientos conmigo, por su tiempo, paciencia, consejos y dedicación a lo largo de todos los años de cursado.

TÍTULO DEL PROYECTO

"Redes anónimas como instrumento para la conformación de un ciber-ataque sobre una infraestructura crítica: Sistema de Radarización de Tráfico Aéreo".

GLOSARIO Y LISTADO DE SÍMBOLOS Y CONVENCIONES

Backup: se refiere a una copia de respaldo de una información determinada.

BD: Base de datos.

Base de datos: es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

Script: es un archivo que tiene como contenido un conjunto de comandos.

Nodo: es una máquina que pertenece a la red Tor y dona parte de su ancho de banda a la misma.

Bridges: son nodos Tor privados, lo cuales sirven como opción para evitar medidas de censura implementadas por un ISP.

ISP: se refiere al proveedor de internet.

Infraestructura crítica: es una infraestructura que en caso de fallar o de concretarse una amenaza puede producir grandes pérdidas económicas, pérdidas o filtraciones de información confidencial, daños a la vida de los ciudadanos e incluso cobrarse muertes humanas.

Proxy: es un servidor (un programa o sistema informático), que sirve de intermediario en las peticiones de recursos que realiza un cliente (A) a otro servidor (C).

Red Anónima: es una red en donde no se conoce la identidad de los interactuantes y miembros de la misma.

Servidor: es una aplicación en ejecución capaz de atender las peticiones de un cliente y devolverle una respuesta en concordancia.

The Onion Router: es una red anónima que trabaja sobre el tráfico http.

TOR: The Onion Router.

ÍNDICE DE FIGURAS

ÍNDICE

1. RESUMEN

Este proyecto atiende a la necesidad de tomar medidas de seguridad con relación a ciber-ataques que se puedan montar desde una red anónima en contra infraestructuras críticas de un país.

Se buscó aplicar los conocimientos adquiridos a lo largo del cursado de la carrera Ingeniería en Informática e incursionar en el campo de la seguridad informática ampliando dichos conocimientos. Tuvo como objetivo principal poner a prueba la capacidad propia de investigación y desarrollo de métodos y soluciones competentes para un determinado problema.

En primer lugar, se realizó una investigación en internet, trabajos publicados y noticias para adquirir conceptos básicos acerca de infraestructuras críticas y redes anónimas que sirvió como marco teórico para las tareas a realizar. Se investigó sobre las redes anónimas más conocidas, se seleccionó una de ellas en base a diferentes criterios, se desarrollaron e implementaron métodos para mitigar ataques provenientes de dicha red y se realizaron diferentes pruebas para validar los mismos.

En conclusión, se pudo observar que en caso de concretarse un ciber-ataque es un punto crítico poder seguir un rastro para encontrar al responsable pero si se realiza desde una red anónima, como TOR (The Onion Router), esto no es posible. Por lo tanto se deben tomar las medidas necesarias para evitar los ataques provenientes de dichas redes.

Los métodos desarrollados han sido efectivos y es una medida más para sumar a la seguridad de las infraestructuras críticas de un país.

2. INTRODUCCIÓN

Este proyecto atiende a la necesidad de tomar medidas de seguridad con relación a ciber-ataques que se puedan montar desde una red anónima en contra infraestructuras críticas de un país. La seguridad informática en este tipo de infraestructuras es muy importante debido a que si una amenaza o riesgo se materializa el impacto es muy grande desde pérdida de datos o robo de información confidencial hasta daños a la vida de los habitantes del país.

El tema tratado fue abordado a causa del interés personal en el área de seguridad informática y por estar relacionado al trabajo a realizar en una ayudantía en el Instituto Universitario Aeronáutico.

Se tomó como base la infraestructura del Sistema de Radarización de Tráfico Aéreo. En un primer momento se pensó en reproducir lo mejor posible la estructura de red del sistema, pero debido a la imposibilidad de que se nos facilite dicha información se decidió tomar una estructura de red, acordada entre grupos de investigación y el profesional a cargo, que pensamos se asemeja a la real a partir de la poca información que dispusimos.

En torno a esa estructura, este proyecto alcanza un punto crítico, que es el contacto de la misma con Internet. Se investigó profundamente la red anónima TOR, se realizaron diversas pruebas y se evaluaron diferentes medidas de seguridad para proteger la infraestructura crítica.

3. OBJETIVO DEL PROYECTO

Mitigar los riesgos asociados a ataques montados sobre redes anónimas contra una infraestructura crítica.

3.1. Objetivos específicos:

- Investigar y asimilar conceptos acerca de infraestructuras críticas
- Investigar y asimilar conceptos acerca de redes anónimas
- Profundizar la investigación sobre una de las redes anónimas, su funcionamiento y el riesgo potencial que representan
- Identificar métodos para mitigar ataques desde la red anónima seleccionada
- Implementar los métodos identificados en el punto anterior
- Integrar conocimientos y la investigación realizada con otros grupos de investigación
- Montar y llevar a cabo diferentes ataques sobre la red anónima seleccionada

Se busca aplicar los conocimientos adquiridos a lo largo del cursado de la carrera Ingeniería en Informática e incursionar en el campo de la seguridad informática ampliando dichos conocimientos.

4. DESTINATARIOS

Los destinatarios de este trabajo de investigación son las áreas de I+D+i del Instituto Universitario Aeronáutico, Sistema de Radarización asociado al Tráfico Aéreo y potencialmente el Sistema de Radarización de la Fuerza Aérea Argentina.

5. BENEFICIOS

Los beneficios que se pretenden obtener están relacionados con la mejora en la seguridad de la infraestructura crítica y de esta forma evitar pérdidas monetarias, accidentes aéreos, confiabilidad de la información del tráfico aéreo y robustez de la infraestructura ante posibles ciber-ataques.

6. ESTUDIO TÉCNICO

Para la investigación se requiere de sólo una computadora, acceso a internet y acceso a papers de la IEEE otorgado por el IUA.

Para recrear la infraestructura crítica y montar los ataques se utilizarán virtualizaciones, sistemas operativos Linux y todas las herramientas brindadas por el software de la red anónima. Se utilizará software open source ya que obtenemos herramientas muy completas y gratuitas para las tareas a realizar.

7. DESARROLLO DEL TRABAJO

7.1. Resumen técnico

En la actualidad, la importancia de las infraestructuras críticas de un país es un blanco claro para ciber-ataques y el ciber-terrorismo, ya que atacar a una de ellas provoca un gran impacto en el país.

Por otro lado, los ataques montados sobre redes anónimas, en las cuales no se conoce, en primera instancia, al autor de los mismos, representan una amenaza inminente para una infraestructura crítica.

Por estos motivos, entre otros, se seleccionó una red anónima (TOR) y se buscaron métodos para poder mitigar los riesgos provenientes de esta red y así proteger a la infraestructura crítica en cuestión. Se reprodujo la parte de la estructura de red que está afectada a vulnerabilidades que se puedan aprovechar con TOR, se realizaron pruebas dejando como evidencia el impacto que produce que un ataque se materialice y se verificó la efectividad de las medidas de seguridad desarrolladas.

7.2. Metodología

El trabajo a realizar se comenzó conjuntamente con la ayudantía en seguridad informática desarrollada en el IUA a partir del 12 de mayo de 2014.

En primer lugar, se realizó una investigación en internet, trabajos publicados y noticias para adquirir conceptos básicos acerca de infraestructuras críticas y redes anónimas que servirá como marco teórico para las tareas a realizar.

Se evaluaron las diferentes redes anónimas y se seleccionó la red TOR.

Se investigaron los métodos para mitigar los ataques provenientes de TOR.

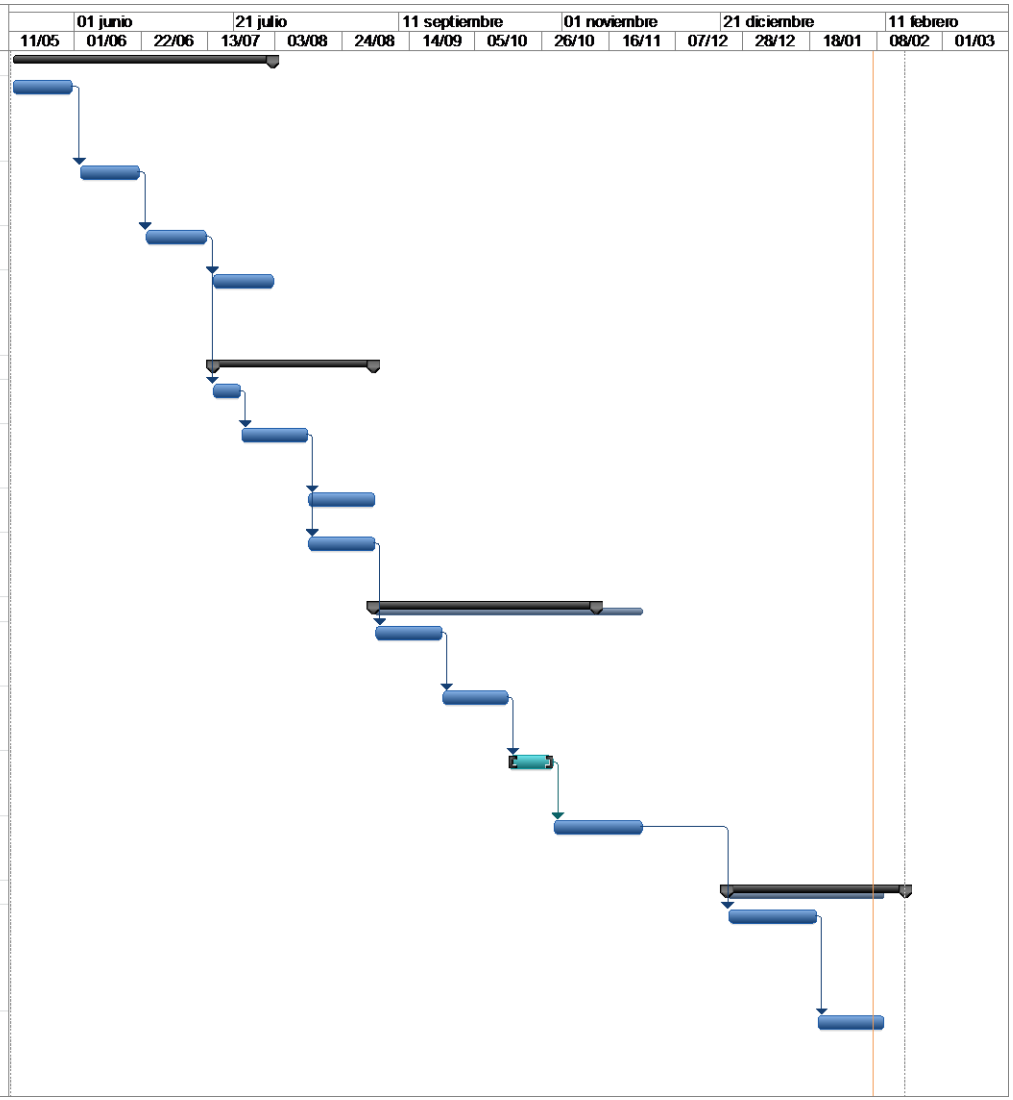
Una vez realizados estos pasos, se implementaron los métodos identificados en una red reducida y se realizaron ataques a modo de pruebas de laboratorio para verificar la efectividad de dichos métodos.

Se integró la investigación con otros grupos de investigación encargados de los ataques y la defensa de la infraestructura crítica, lo cual permitió enriquecer los conocimientos hasta el momento adquiridos y avanzar con las tareas pendientes.

Finalmente se conformó dicha infraestructura crítica, enfocándose en la parte que afecta directamente a este proyecto, se montaron ataques para mostrar el impacto de su materialización, se verificó la efectividad de las medidas de seguridad empleadas y se llegó a una conclusión.

7.3. Actividades realizadas

Id	Nombre de tarea	Duración	Predecesoras	Comienzo	Fin	Nombres de los recursos
1	**Investigación**	**60 días**		**lun 12/05/14**	**vie 01/08/14**	
2	Buscar y asimilar información acerca de Infraestructuras Críticas	15 días		lun 12/05/14	vie 30/05/14	
3	Buscar y asimilar información acerca de Redes Anónimas	15 días	2	lun 02/06/14	vie 20/06/14	
4	Investigar una red anónima particular	15 días	3	lun 23/06/14	vie 11/07/14	
5	Investigar métodos para mitigar ataques provenientes de la red anónima	15 días	4	lun 14/07/14	vie 01/08/14	
6	**Desarrollo**	**37 días**		**lun 14/07/14**	**mar 02/09/14**	
7	Construir una red de prueba virtualizada	7 días	4	lun 14/07/14	mar 22/07/14	
8	Llevar a cabo métodos de mitigación	15 días	7	mié 23/07/14	mar 12/08/14	
9	Realizar ataques de prueba	15 días	8	mié 13/08/14	mar 02/09/14	
10	Evaluar la efectividad de los métodos utilizados	15 días	8	mié 13/08/14	mar 02/09/14	
11	**Integración**	**50 días?**		**mié 03/09/14**	**mar 11/11/14**	
12	Integrar investigaciones con otros equipos	15 días	10	mié 03/09/14	mar 23/09/14	
13	Construir y simular la infraestructura crítica	15 días	12	mié 24/09/14	mar 14/10/14	
14	Crear un servidor web con una aplicación de prueba	10 días	13	mié 15/10/14	mar 28/10/14	
15	Montar diferentes ataques sobre la red anónima	20 días	14	mié 29/10/14	mar 25/11/14	
16	**Evaluación**	**40 días**		**mar 23/12/14**	**lun 16/02/15**	
17	Evaluar la efectividad de los métodos utilizados y llegar a una conclusión	20 días	15	mar 23/12/14	lun 19/01/15	
18	Redacción de informe de Trabajo Final de Grado	15 días	17	mar 20/01/15	lun 09/02/15	



7.4. Dificultades que se han presentado

La principal dificultad que se presentó fue la imposibilidad de acceso a la infraestructura real del sistema de radarización de tráfico aéreo. Esto produjo un retraso en las tareas programadas del trabajo final. Para superar esto, en conjunto con otros grupos de investigadores y el tutor correspondiente, se planteó una infraestructura ficticia pero que se asemejaría a la real.

7.5. Resultados alcanzados

- Se adquirieron conceptos sobre infraestructuras críticas y redes anónimas.
- Se obtuvieron datos analíticos sobre la actividad de los nodos de TOR en un período de tiempo determinado.
- Se desarrollaron métodos para mitigar ataques montados desde TOR.
- Se publicó un paper en el CoNaISII sobre el tema tratado, la investigación, el análisis y las pruebas realizadas hasta el momento.
- Se conformó exitosamente la infraestructura crítica planteada.
- Se verificó la efectividad de los métodos de mitigación desarrollados.
- Se pudo observar un comportamiento de rotación de identidad por parte de TOR para intentar eludir los métodos de mitigación desarrollados.

8. PROYECCIÓN DE COSTOS DE OPERACIÓN Y MANTENIMIENTO

Una vez puesto en funcionamiento el proyecto solo requerirá mantenimiento al evolucionar las redes anónimas o los ataques analizados. En tal caso, se deberá repetir el proceso de investigación y mejorar los métodos propuestos o buscar nuevos.

9. ESTUDIO AMBIENTAL

El presente proyecto no presenta un impacto importante para el medio ambiente, ya que prácticamente solo se consumirá energía eléctrica.

10. INTRODUCCIÓN A REDES ANÓNIMAS

Históricamente, el objetivo principal de estas redes era para evitar la censura "política" de algunos países y permitir la libertad de expresión en Internet a través del anonimato de los usuarios. Sin embargo, a través del tiempo se le ha dado otro tipo de uso en actividades ilegales de venta de armas y drogas, pornografía infantil, documentos falsos, publicación de documentos confidenciales de diferentes gobiernos, etc.

11.1. I2P

Invisible Internet Project (Proyecto de Internet invisible) es un software que ofrece una capa de abstracción para comunicaciones entre ordenadores, permitiendo así la creación de herramientas y aplicaciones de red con un fuerte anonimato.

11.1.1. Funcionamiento

La red I2P está basada en el concepto de túneles entrantes y salientes, lo cual ofrece facilidad para la adaptación de programas preexistentes a la red I2P. Cada túnel está compuesto por una secuencia de nodos padres, los cuales transportan la información en un sentido unidireccional.

11.2. Freenet

Es una red de distribución de información descentralizada y resistente a la censura. El objetivo de Freenet es almacenar documentos y permitir su acceso posterior por medio de una clave asociada, impidiendo que sea posible la censura de los documentos y ofreciendo anonimato tanto al usuario que publica el documento como al que lo descarga. Como parte del apoyo a la libertad de sus usuarios, Freenet es software libre y ha estado bajo continuo desarrollo desde el año 2000, y aunque todavía no ha sido liberada una posible versión 1.0, las versiones actuales son completamente funcionales.

11.2.1. Funcionamiento

Freenet se diseñó como una red P2P no estructurada de nodos no jerarquizados que se transmiten mensajes y documentos entre ellos. Los nodos pueden funcionar como nodos finales, desde donde empiezan las búsquedas de documentos y se presentan al usuario, o como nodos intermedios de enrutamiento. Cada nodo aloja documentos asociados a claves y una tabla de enrutamiento que asocia nodos con un historial de su desempeño para adquirir diferentes claves.

Para encontrar un documento en la red conocida con una clave que lo describe, un usuario envía un mensaje a un nodo solicitando el documento y proveyéndolo con la clave. Si el documento no se encuentra en la base de datos local, el nodo selecciona a un vecino de su tabla de enrutamiento que cree que será capaz de localizar la clave más rápidamente y le pasa la petición, recordando quién envió el mensaje para poder deshacer después el camino. El nodo al que se pasó la petición repite el proceso hasta que se encuentra un nodo que guarda el documento asociado a la clave o la petición pasa por un número máximo de nodos, conocido como el valor de tiempo de vida. Ninguno de los nodos intermedios sabe si el nodo anterior de la cadena fue el originador de la petición o un simple enrutador. Al deshacer el camino, ninguno de los nodos puede saber si el nodo siguiente es el que efectivamente tenía el documento o era otro enrutador. De esta manera, se asegura el anonimato tanto del usuario que realizó la petición como del usuario que la respondió.

Los nodos intermedios pueden elegir mantener una copia temporal del documento en el camino. Además de ahorrar tiempo y ancho de banda en peticiones futuras del mismo documento, esta copia ayuda a impedir la censura del documento, ya que no existe un "nodo fuente", y dificulta adivinar qué usuario publicó originalmente el documento.

11.3. The Onion Router

Tor, es un proyecto cuyo objetivo principal es el desarrollo de una red de comunicaciones distribuida de baja latencia y superpuesta sobre internet en la que el encaminamiento de los mensajes intercambiados entre los usuarios no revela su identidad, es decir, su dirección IP (anonimato a nivel de red) y que, además, mantiene la integridad y el secreto de la información que viaja por ella. Por este motivo se dice que esta tecnología pertenece a la llamada darknet o red oscura también conocida con el nombre de deep web o web profunda.

Tor propone el uso de encaminamiento de cebolla de forma que los mensajes viajen desde el origen al destino a través de una serie de routers especiales llamados 'routers de cebolla' (en inglés onion routers).

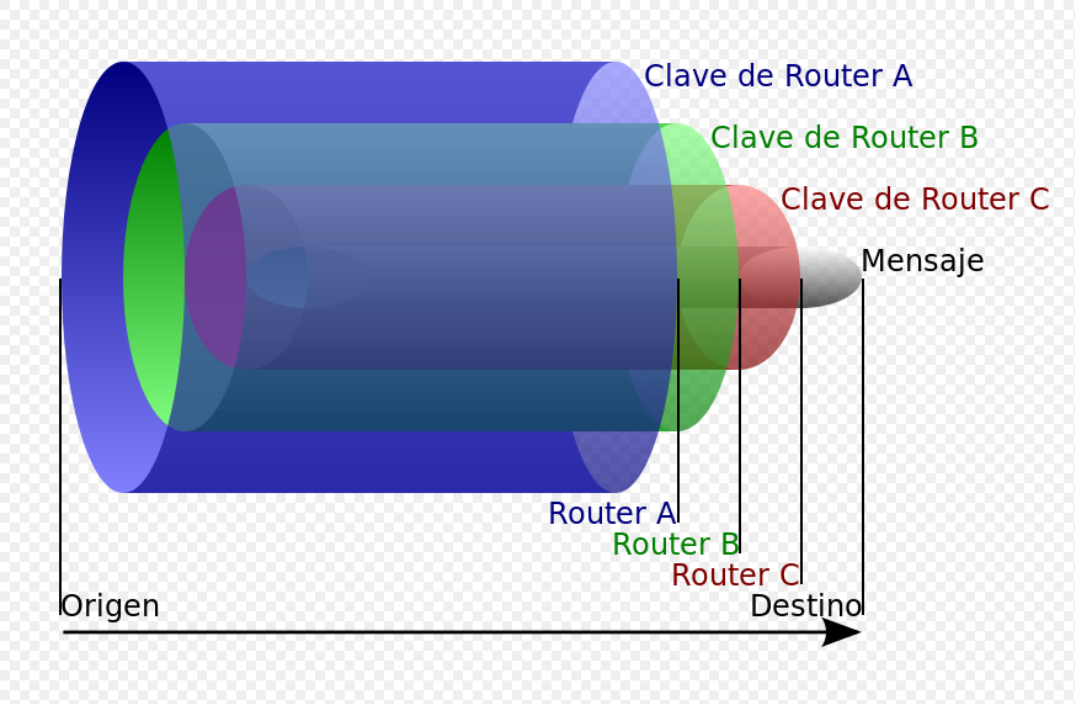


Figura 1: Capas de cebolla de un mensaje de Tor

No es una red entre iguales (peer-to-peer) ya que por un lado están los usuarios de la red y por otro lado los encaminadores del tráfico, algunos de los cuales hacen una función de servicio de directorio.

La red funciona a partir de un conjunto de organizaciones e individuos que donan su ancho de banda y poder de procesamiento.

11.3.1. Funcionamiento

Tor sólo permite anonimizar tráfico TCP y nunca transmite tráfico de usuarios a través de más de dos nodos. Entonces nuestro camino está dado por un nodo de entrada, un nodo intermedio y un nodo de salida.

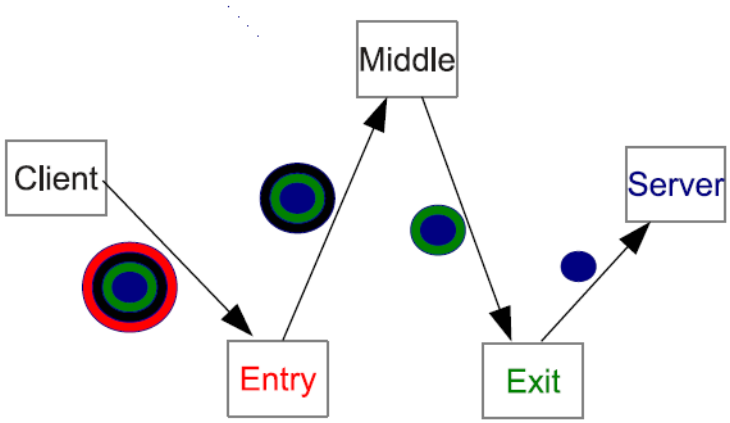


Figura 2: Camino típico por nodos Tor

El primer nodo (entrada) conoce la fuente, pero no el destino, el último nodo (salida) sabe el destino, pero no la fuente y cualquier nodo intermedio no sabe ni el origen ni el de destino.

Nos concentraremos en Tor debido a que está montada sobre la web y es una de las redes anónimas más famosas y utilizadas.

11.3.2. Métricas

En el sitio de Tor (https://www.torproject.org/) podemos encontrar diferentes métricas y para cada una aplicar diferentes filtros para obtener la información deseada. En el primer gráfico se puede observar la cantidad de usuarios conectados a Tor a través del tiempo y podemos ver que actualmente están conectados alrededor de 2.300.000 usuarios.

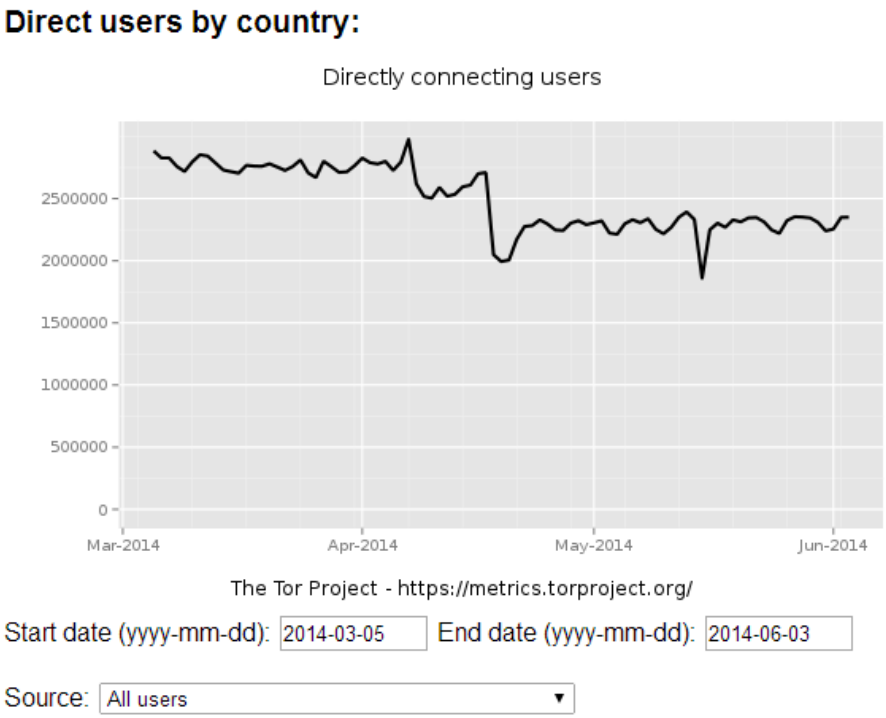


Figura 3: Cantidad de usuarios de Tor en un período determinado

En este gráfico podemos ver los usuarios conectados en Argentina y se observa que actualmente el número ronda los 60.000 usuarios.

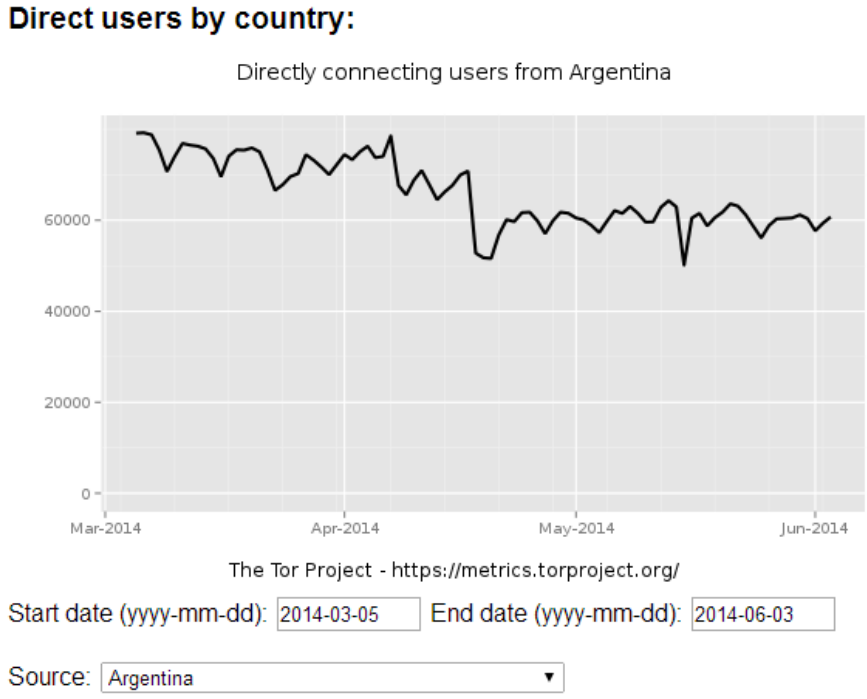


Figura 4: Cantidad de usuarios de Tor de Argentina en un período determinado

En la siguiente tabla se puede apreciar los 10 países con más usuarios conectados por día entre el 8 de marzo de 2014 y el 6 de junio de 2014. Vemos que Argentina se encuentra en el 10mo puesto.

Top-10 countries by directly connecting users:

Start date (yyyy-mm-dd): 2014-03-08 End date (yyyy-mm-dd): 2014-06-06

Update table

Country	Mean daily users
United States	319379 (12,88 %)
Germany	216013 (8,71 %)
France	156273 (6,30 %)
Brazil	151966 (6,13 %)
Spain	115318 (4,65 %)
Italy	103119 (4,16 %)
United Kingdom	91553 (3,69 %)
Russia	88865 (3,58 %)
Poland	79801 (3,22 %)
Argentina	65423 (2,64 %)

Figura 5: Top 10 de países con más usuarios de Tor en un período determinado

La siguiente tabla se desprende de un trabajo de investigación sobre el tráfico de la red Tor y podemos ver las categorías de los sitios web más accedidos por los usuarios.

Rank	Category	Percentage
1	Search Engines/Portals	14.45%
2	Pornography	11.50%
3	Computers/Internet	11.45%
4	Social Networking	9.52%
11	Blogs/Web Communications	2.26%
13	StreamingMedia/MP3	1.82%
14	Software Downloads	1.66%
36	Hacking	0.3%
40	Political	0.18%
42	Illegal/Questionable	0.15%
52	IllegalDrugs	0.06%

Figura 6: Principales tráficos identificados en la red Tor

11.3.3. Vulnerabilidades

Para romper el anonimato de Tor, es suficiente para observar el tráfico de los nodos de entrada y salida. De hecho, si se observa el tráfico de los nodos de entrada y salida, un análisis de tráfico simple puede confirmar fácilmente quién se comunica con quién. Dado que los nodos de reenvío del tráfico de Tor son máquinas de los usuarios voluntarios, un ataque muy popular en la literatura consiste en inyectar un cierto número de nodos maliciosos en la red Tor y luego esperar a que el cliente Tor utilice dos de

ellos como los nodos de entrada y salida. Existen diversas variantes de este ataque en la literatura. La principal limitación de este ataque es su escalabilidad: con un par de cientos de relés el ataque es eficiente, sin embargo, con los 3.000 aparatos de la red Tor actual (2013), el ataque ya no es escalable.

Debilidades del encaminamiento de cebolla:

Análisis de tiempos (en inglés Timing analysis): Analizando el flujo de mensaje que pasan a través de un nodo poco cargado podemos deducir la correspondencia entre ciertos mensajes de entrada y otros de salida. Para evitar esto se propone el uso de mensajes del mismo tamaño, tráfico de relleno y la introducción tiempos de espera artificiales que lo dificulten.

Ataques para la denegación de servicio (en inglés Denial Of Sevices attacks): En este tipo de redes es fácil desarrollar ataques de denegación de servicio por ejemplo forzando a muchos routers a realizar una fuerte cantidad de operaciones criptográficas o agotando el ancho de banda. Para evitar este tipo de ataques de forma radical se han propuesto sistemas en los que se establece alguna forma de 'moneda digital' que los clientes tienen que 'pagar' para usar los servicios del sistema. Por ejemplo se podrían usar sistemas que obligaran a los usuarios a realizar un esfuerzo importante y que fueran fáciles de verificar.

Ataques de replay: Los sistemas con encaminamiento de cebolla son vulnerables a ataques de replay que se basan en capturar mensajes y luego los reinyectan en la red con el objetivo de sobrecargarla y que deje de funcionar (ataque de denegación de servicio). Para evitar este tipo de ataques es habitual que los routers detecten cuando un paquete ya ha sido procesado (y por tanto descarten ese paquete) y que los propios mensajes tengan un tiempo de validez que una vez agotado permita que los routers eliminen esos mensajes.

Ataques de intersección (en inglés Intersection attacks): Este ataque se basa en obtener una colección de conjuntos de nodos que sabe que contienen el iniciador de la comunicación. A partir de este conjunto podemos sacar conclusiones válidas simplemente hallando la intersección de estos conjuntos. Si la intersección es sólo un nodo entonces se puede estar seguro de quién es el iniciador. Para que los distintos conjuntos de nodos que vamos obteniendo sean distintos, y por tanto tenga sentido la hacer la intersección, podemos analizar los nodos que se desconectan o entran en la red mientras cierto circuito continúa funcionando sin ningún tipo de problema. El coste de este tipo de ataques es significante si el tamaño de la red es grande, pero puede ser factible en algunos escenarios.

Ataques de predecesor (en inglés Predecessor attacks): Este tipo de ataques se basan en dos suposiciones:

- Hay una conexión recurrente entre un iniciador y un destinatario que se intercambian mensajes.
- En los mensajes transportados hay información que puede ser utilizada por el atacante para identificar esa conexión recurrente.

Podemos justificar estas suposiciones basándonos en cómo se comportan las aplicaciones que usan ciertos protocolos. Por ejemplo cuando usamos HTTP, FTP, SSH, Telnet o IRC normalmente nos conectamos recurrentemente al mismo servidor.

En general los routers de una red con encaminamiento de cebolla no tienen conexiones totalmente estables. Cuando un nodo se desconecta, cualquier circuito en el que participara ese nodo se ve afectado y tiene que volver a configurarse (en inglés se le llama chain reformation). A este evento le vamos a llamar reset. Al periodo de tiempo entre dos resets lo vamos a llamar una ronda.

Los ataques de predecesor se basan en lo siguiente: Si un nodo comprometido intercepta un mensaje, el nodo predecesor tendrá una probabilidad más alta de ser el nodo iniciador comparado con otros nodos. Por tanto para realizar el ataque el atacante estudia las rondas en las cuales se envían mensajes que son parte del flujo que estamos estudiando y que pasan a través de nodos comprometidos (los cuales almacenan de forma pasiva toda la información para poder hacer luego el análisis). A partir de ahí se establece el nodo predecesor con probabilidad más alta.

Ataque basado en capturar el tráfico de los nodos de salida: Los nodos de salida (los últimos de la cadena de routers) tienen un acceso completo al contenido que se transmite desde el origen al destino. Basándose en esto ha habido ataques que han capturado información privilegiada. Este tipo de ataques pueden ser resueltos empleando cifrado extremo a extremo, por ejemplo usando SSL.

11. CONFIGURACIÓN INICIAL PROXY-FIREWALL

12.1. Introducción

Ante la problemática planteada de recibir ataques que provengan de redes anónimas vamos a implementar, en el sistema operativo Fedora 18, Squid proxy y reglas del firewall Iptables para evitar el tráfico proveniente de estas redes.

Vamos a trabajar con la siguiente topología:

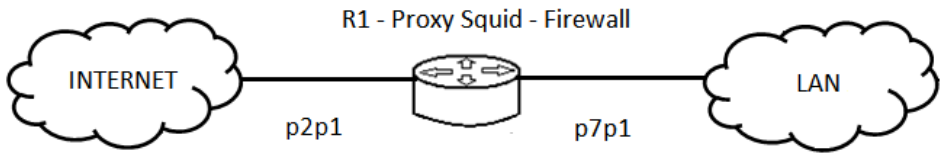


Figura 7: Esquema de red para pruebas de laboratorio

INTERNET:
IP: 192.168.1.0
Mask: /24
LAN:
IP: 192.168.2.0
Mask: /24
p2p1:
IP: 192.168.1.150

Mask: /24

p7p1:

IP: 192.168.2.150
Mask: /24

12.2. Instalación de Squid Proxy e Iptables Firewall

Para instalar Squid ejecutamos el comando:

 # yum –y install squid

Si tenemos éxito vemos algo similar a lo siguiente:

```
Installed:
  squid.i686 7:3.2.13-1.fc18

Complete!
```

Luego habilitamos el servicio para que cargue al inicio del sistema y corremos el servicio:

 # systemctl enable squid.service
 # systemctl start squid.service

Para trabajar con Iptables, primero debemos deshabilitar firewalld, por lo que ejecutamos los siguientes comandos:

 # systemctl stop firewalld.service
 # systemctl disable firewalld.service

Luego debemos instalar los paquetes necesarios de la siguiente manera:

#yum –y install iptables-services system-config-firewall system-config-firewall-tui

Si la instalación es correcta obtenemos el siguiente mensaje:

```
Installed:
  iptables-services.i686 0:1.4.16.2-5.fc18
  system-config-firewall.noarch 0:1.2.29-8.fc18
  system-config-firewall-tui.noarch 0:1.2.29-8.fc18

Dependency Installed:
  iptables.i686 0:1.4.16.2-5.fc18
  system-config-firewall-base.noarch 0:1.2.29-8.fc18
```

Habilitamos iptables para que se cargue al inicio del sistema e iniciamos el servicio:

 # systemctl enable iptables.service
 # systemctl enable ip6tables.service
 # systemctl start iptables.service
 # systemctl start ip6tables.service

Si tenemos problemas para iniciar el servicio debemos ejecutar el comando **# setup** e ir a **Firewall configuration**:

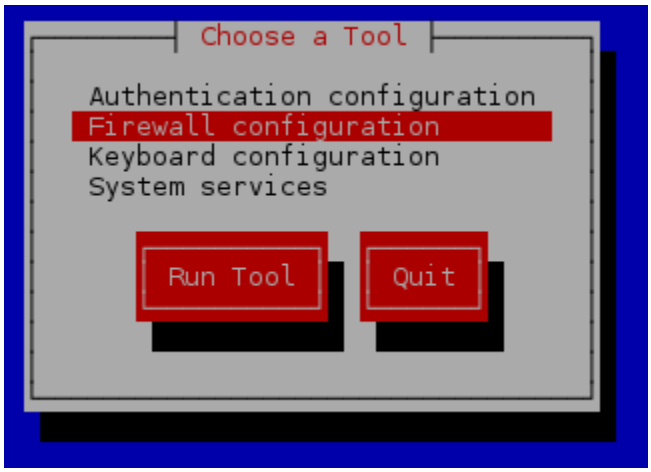


Figura 8: Configuración firewall 1

Luego con la barra espaciadora tildamos la opción **Enabled** y presionamos **OK:**

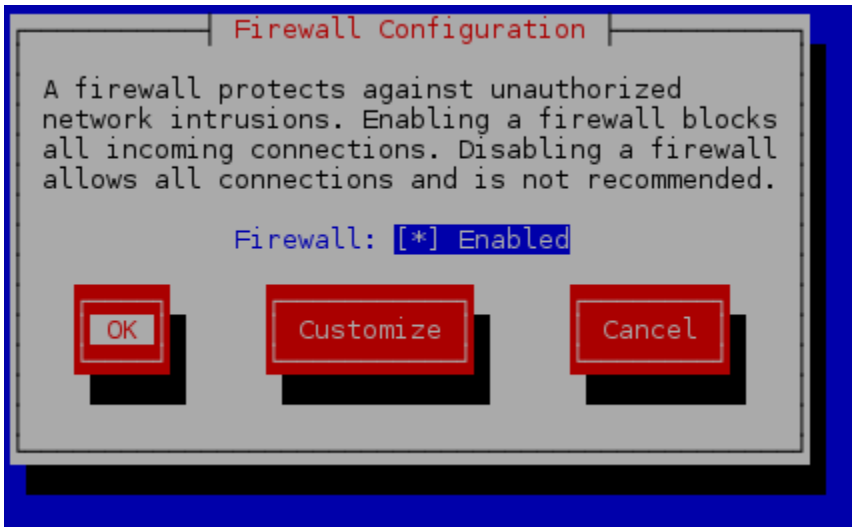


Figura 9: Configuración firewall 2

12.3. Configuración de redes

Establecemos la configuración IP en una máquina que estará en la red de **Internet**:

```
HWADDR=08:00:27:0E:A4:F8
TYPE=Ethernet
#BOOTPROTO=dhcp
BOOTPROTO=static
IPADDR=192.168.1.110
NETMASK=255.255.255.0
GATEWAY=192.168.1.1
DEFROUTE=yes
PEERDNS=yes
PEERROUTES=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
NAME=p2p1
UUID=44bc5e80-4f6d-42d5-b399-9bbb1ab71d21
ONBOOT=yes
```

Figura 10: Configuración IP PC Internet

Configuración de ruta estática para la red LAN 192.168.2.0/24 en el archivo **/etc/sysconfig/network-scripts/route-p2p1**:

```
default 192.168.1.1 dev p2p1
192.168.2.0/24 via 192.168.1.150 dev p2p1
```

Configuramos un equipo dentro de la red **LAN**:

```
HWADDR=08:00:27:EC:63:7F
TYPE=Ethernet
#BOOTPROTO=dhcp
BOOTPROTO=static
IPADDR=192.168.2.105
NETMASK=255.255.255.0
GATEWAY=192.168.2.150
DEFROUTE=yes
PEERDNS=no
PEERROUTES=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
NAME=p2p1
UUID=a29aae94-39cf-4aaa-a975-16a9b23b1715
ONBOOT=yes
```

Figura 11: Configuración IP PC LAN

Luego, creamos el archivo **/etc/sysconfig/network-scripts/route-p2p1** con la siguiente información de rutas:

```
default 192.168.2.150 dev p2p1
192.168.1.0/24 via 192.168.2.150 dev p2p1
```

En la máquina que hará el papel de router configuramos la interfaz **p2p1** de la siguiente manera:

```
HWADDR=08:00:27:F8:4B:49
TYPE=Ethernet
#BOOTPROTO=dhcp
BOOTPROTO=static
IPADDR=192.168.1.150
NETMASK=255.255.255.0
GATEWAY=192.168.1.1
DEFROUTE=yes
#PEERDNS=yes
PEERDNS=no
PEERROUTES=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
NAME=p2p1
UUID=3b6dc792-690d-481d-b788-fe4ff0d7da2b
ONBOOT=yes
```

Figura 12: Configuración IP Router p2p1

Y luego configuramos la interfaz **p7p1**:

```
HWADDR=08:00:27:51:B5:7F
TYPE=Ethernet
#BOOTPROTO=dhcp
BOOTPROTO=static
IPADDR=192.168.2.150
NETMASK=255.255.255.0
GATEWAY=192.168.1.1
DEFROUTE=yes
PEERDNS=no
PEERROUTES=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
NAME=p7p1
UUID=a534a4f1-a19a-4df0-abab-e940937a7621
ONBOOT=yes
```

Figura 13: Configuración IP Router p7p1

Habilitamos ip forward: accedemos al archivo **/usr/lib/sysctl.d/00-system.conf** y modificamos la línea "**net.ipv4.ip_forward=1**". Esto permite el reenvío de paquetes que pasan por el equipo.

```
# Controls IP packet forwarding
net.ipv4.ip_forward = 1
```

Figura 14: Configuración IP Forward Router

12.4. Configuración del proxy Squid

Editamos el archivo /etc/squid/squid.conf:

Incluimos nuestra red LAN en la Lista de Control de Acceso (ACL – Access Control List):

```
acl localnet src 192.168.2.0/24 # LAN
```

Damos acceso por protocolo **http** a nuestra **localnet** definida anteriormente:

```
# Example rule allowing access from your local networks.
# Adapt localnet in the ACL section to list your (internal) IP networks
# from where browsing should be allowed
http_access allow localnet
```

Figura 15: Permitir acceso http a acl localnet

Para hacer transparente el proxy editamos la línea: **http_port 3128 transparent**. Esto significa que nuestra red LAN tendrá acceso a INTERNET pero desde INTERNET no se puede ver a nuestra red LAN.

```
# Squid normally listens to port 3128
#http_port 3128
http_port 3128 transparent
```

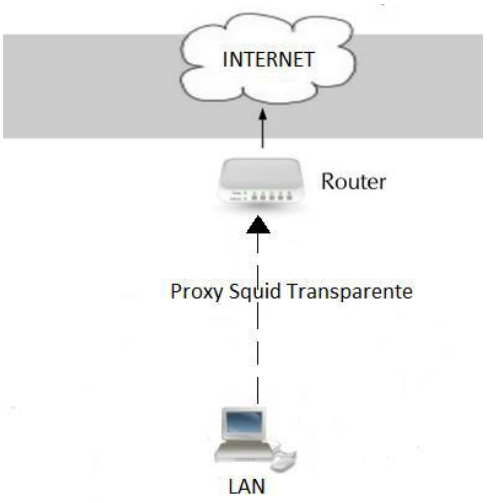


Figura 16: Configuración Proxy Transparente

12.5. Configuración del firewall Iptables - Squid

Configuramos el firewall Iptables para trabajar con Squid a través del siguiente script.

```
#!/bin/bash
#
iptables -F
iptables -X
iptables -F -t nat
iptables -X -t nat
iptables -t nat -A POSTROUTING -s 192.168.2.0/24 -o p2p1 -j SNAT --to 192.168.1.150
iptables -t nat -A PREROUTING -i p7p1 -p tcp --dport 80 -j DNAT --to 192.168.1.150:3128
iptables -t nat -A PREROUTING -i p2p1 -p tcp --dport 80 -j REDIRECT --to-port 3128
```

Figura 17: Script inicial iptables_squid.sh

Primero, eliminamos las reglas iptables existentes para trabajar en limpio. Por otro lado, en este caso redireccionamos al puerto transparente el tráfico que pasa por el puerto 80, en adelante trabajaremos con el tráfico de otro puerto si fuera necesario.

12.6. Configuración del firewall Iptables - TOR

A través del siguiente script se encontró la forma de bloquear el tráfico proveniente de la red anónima TOR. Ya sabemos que TOR nos asigna una IP dentro de su red. Entonces, si podemos bloquear cada uno de los nodos de TOR, no se permitirá el tráfico proveniente de los mismos y si colocamos el script en cada equipo de trabajo, tampoco se podrá acceder a la red TOR ya que no nos podrá asignar ninguna IP "anónima".

Ahora bien, el script consiste en obtener un listado de los nodos TOR y bloquearlos a través de reglas iptables. La página de internet http://torstatus.blutmagie.de/ nos proporciona, de manera accesible, un listado actualizado de los nodos que necesitamos. A continuación podemos ver las líneas de este script:

```
#!/bin/bash
#

cnt=0

wget http://torstatus.blutmagie.de/ip_list_all.php/Tor_ip_list_ALL.csv -O Tor_ip_list_ALL

for nodo in $(cat Tor_ip_list_ALL)
do
        iptables -A FORWARD -s $nodo -j DROP
        ((cnt++))
done

echo $cnt nodos bloqueados.
```

Figura 18: Script inicial iptables_tor.sh

A través del comando wget obtenemos un archivo de la página web con el listado de los nodos TOR y lo guardamos en el archivo Tor_ip_list_ALL. Luego, para cada IP almacenada en este archivo, cargamos una regla iptables para que se eliminen los paquetes que se tengan que reenviar y provengan de esas direcciones. Se utiliza un contador para saber la cantidad de nodos que hemos bloqueado.

12.7. Mejora de Script y black list de nodos TOR

En este punto veremos cómo mejorar el script anterior para bloquear los nodos TOR y guardar en un archivo todas las IP de los mismos. En este caso, guardaremos las direcciones IP de los nodos, obtenidas en una primera ejecución del script, en nuestra black list. Como la lista de nodos que obtenemos son los que están activos en ese momento, y algunos, por diferentes motivos, pueden estar inactivos, en las próximas ejecuciones del script comparamos la nueva lista con la black list y se guardan las nuevas IP solamente. De esta forma, podremos cubrir los nodos que representarían un "hueco" de seguridad ya que pueden estar fuera de servicio momentáneamente (lo que significa que no estarán presentes en la lista que descarguemos en ese momento) y en el siguiente momento vuelven a estar activos (por lo que esa IP se le puede asignar a un cliente y en caso de ser más que uno se podrán hacer los saltos necesarios para lograr el anonimato, teniendo esta posibilidad una muy baja probabilidad).

Guardamos la primera lista en el archivo **Tor_black_list** y luego actualizaremos el mismo con las nuevas direcciones en la lista de **Tor_ip_list_ALL**. A continuación podemos ver el script modificado:

```
#!/bin/bash
#

total=0
nuevos=0

# Descarga de la lista actual de nodos y almacenamiento en el archivo Tor_ip_list_ALL
wget http://torstatus.blutmagie.de/ip_list_all.php/Tor_ip_list_ALL.csv -O Tor_ip_list_ALL

# Ordenamiento de la lista del archivo Tor_ip_list_ALL
sort -t . -k 1,1n -k 2,2n -k 3,3n -k 4,4n Tor_ip_list_ALL -o Tor_ip_list_ALL

# Ordenamiento de la lista del archivo Tor_black_list
sort -t . -k 1,1n -k 2,2n -k 3,3n -k 4,4n Tor_black_list -o Tor_black_list

# Almacenamiento de las nuevas IP en el archivo Tor_black_list
# y carga de nuevas reglas iptables
for nodo in $(diff Tor_black_list Tor_ip_list_ALL | grep -E '> ([0-9]{1,3}\.){3}[0-9]{1,3}' | sed 's/^> *//g')
do
        echo $nodo >> Tor_black_list
        iptables -A FORWARD -s $nodo -j DROP
        ((nuevos++))
done

# Nuevo ordenamiento de la lista del archivo Tor_black_list
sort -t . -k 1,1n -k 2,2n -k 3,3n -k 4,4n Tor_black_list -o Tor_black_list

for linea in $(cat Tor_black_list)
do
        ((total++))
done

echo Nuevos nodos bloqueados: $nuevos
echo Total nodos bloqueados: $total
```

Figura 19: Script inicial iptables_tor_black_list.sh

Con el comando **sort** ordenamos de menor a mayor las diferentes listas para poder compararlas y encontrar los nuevos nodos. Con el comando **diff** comparamos las dos listas, Tor_black_list y Tor_ip_list_ALL, y obtenemos todas las diferencias. Lo que nos devuelve diff lo procesamos con grep para obtener solo las IP que no estén en Tor_black_list. Y por último, con el comando **sed** eliminamos los caracteres del principio de la línea que no nos interesan para obtener la dirección IP limpia.

Luego guardamos el nuevo nodo en Tor_black_list (echo $nodo >> Tor_black_list), cargamos la regla iptables correspondiente e incrementamos el contador **nuevos**. Estos pasos se repiten para cada nuevo nodo.

Por otro lado, ordenamos nuevamente la lista Tor_black_list, con los nuevos nodos, para cualquier otro uso que le podamos dar.

Finalmente, contamos la cantidad total de nodos almacenados en Tor_black_list y mostramos los resultados por línea de comandos.

A continuación, podemos ver la ejecución de nuestro script y los resultados obtenidos.

```
[root@192 ~]# ./iptables_tor_black_list.sh
--2014-08-02 02:23:22--  http://torstatus.blutmagie.de/ip_list_all.php/Tor_ip_list_ALL.csv
Resolviendo torstatus.blutmagie.de (torstatus.blutmagie.de)... 192.251.226.204, 2a02:cbf0:10:101::1:8324
Conectando con torstatus.blutmagie.de (torstatus.blutmagie.de)[192.251.226.204]:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: no especificado [application/force-download]
Grabando a: “Tor_ip_list_ALL”

    [    <=>                                                                 ] 85.070      86,1KB/s   en 1,0s

2014-08-02 02:23:29 (86,1 KB/s) - “Tor_ip_list_ALL” guardado [85070]

Nuevos nodos bloqueados: 168
Total nodos bloqueados: 6196
```

Figura 20: Ejecución script iptables_tor_black_list.sh

Nota: también se creó un script similar a **iptables_tor.sh** llamado **tor_black_list.sh** en el cual se cargan las reglas iptables correspondientes a todos los nodos de **Tor_black_list**.

```
#!/bin/bash
#

cnt=0

for nodo in $(cat Tor_black_list)
do
        iptables -A FORWARD -s $nodo -j DROP
        ((cnt++))
done

echo $cnt nodos bloqueados.
```

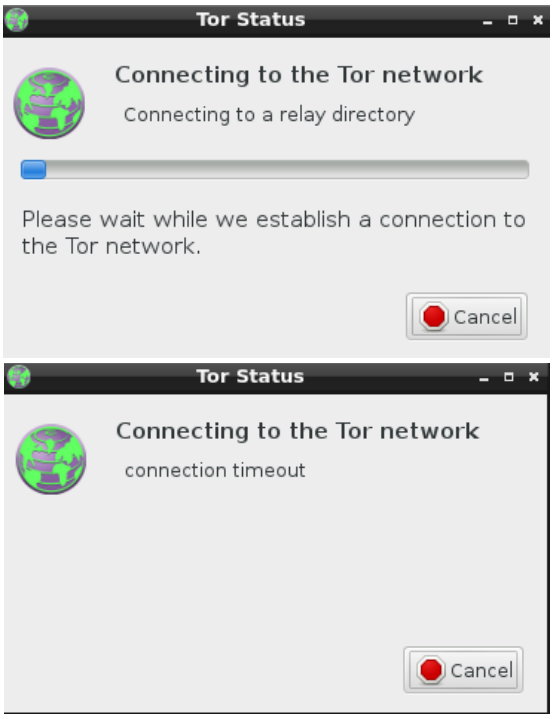
Figura 21: Script tor_black_list.sh

12.8. Evidencias

Después de ejecutar los scripts **iptables_squid.sh**, **iptables_tor.sh** e **iptables_tor_black_list.sh**, vemos que podemos ingresar a internet a través de un navegador común, gracias a que con Squid e Iptables hicimos NAT de nuestra red LAN hacia Internet.

Figura 22: Acceso a Internet desde LAN

Gracias a que bloqueamos todos los nodos de TOR cuando nos queremos conectar como usuarios a la red, se queda intentando establecer dicha conexión pero no lo logra y podemos ver los mensajes de falla.



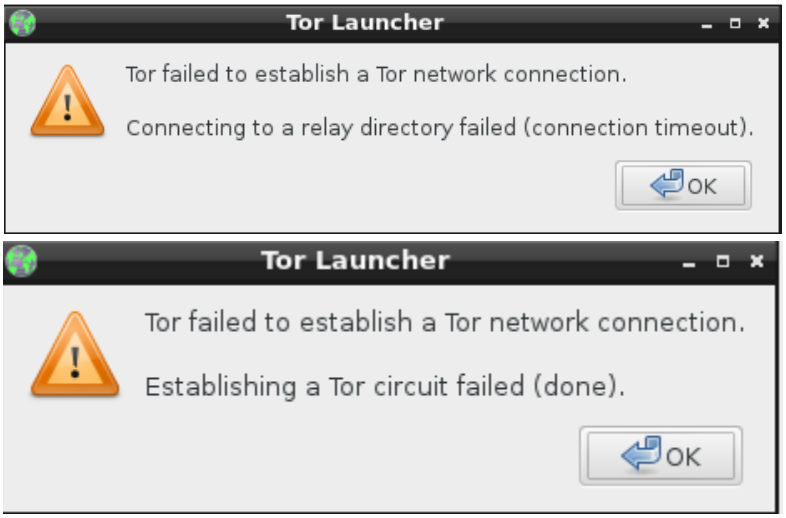
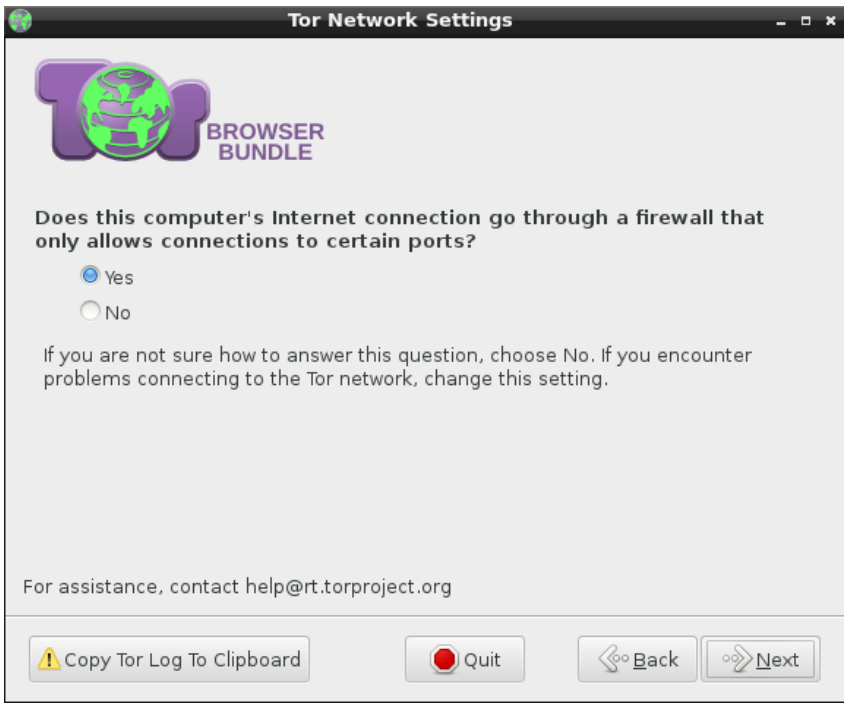


Figura 23: Imposibilidad de conexión con Tor desde LAN

Ahora vamos a poner a prueba nuestro bloqueo con algunas herramientas ofrecidas por TOR. Primero vamos a intentar burlar el firewall a través de diferentes puertos.



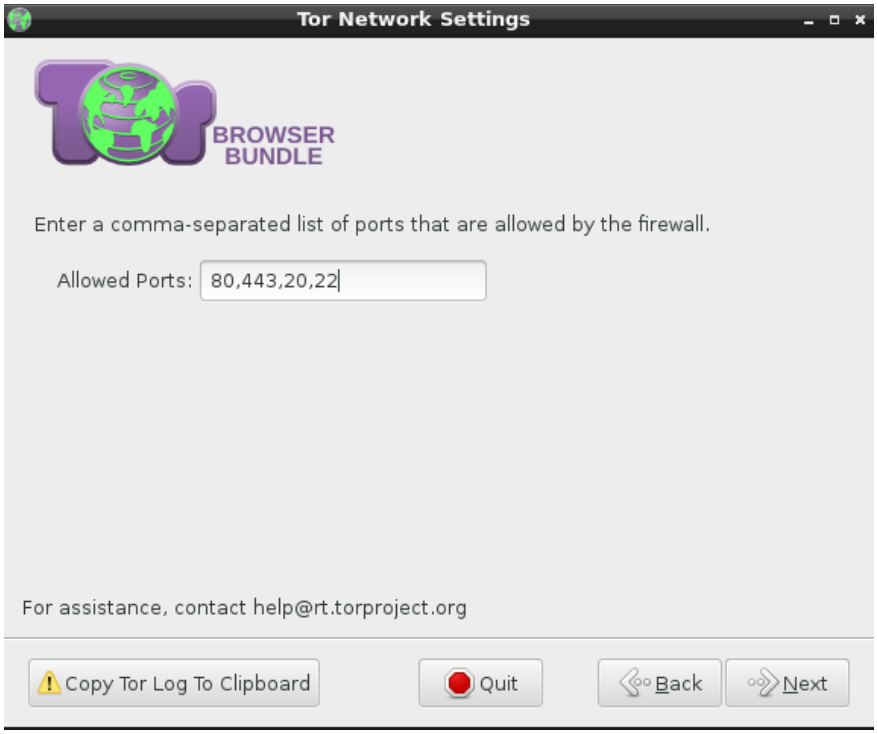
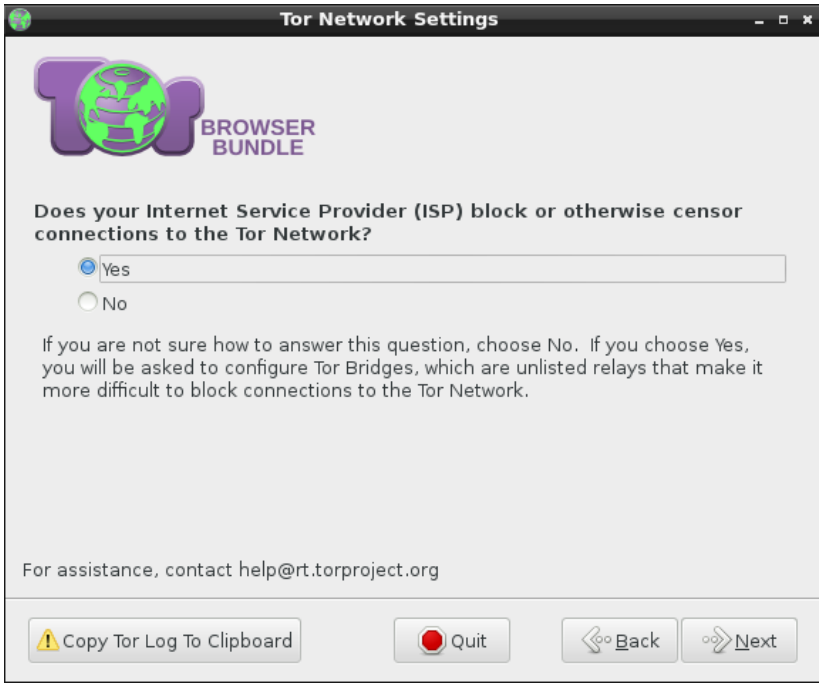


Figura 24: Configuración para evitar reglas del firewall local

Volvimos a obtener el mismo resultado y lo intentamos a través de los puentes que nos ofrece la aplicación. Esto se utiliza cuando los proveedores de internet bloquean algunos nodos TOR.



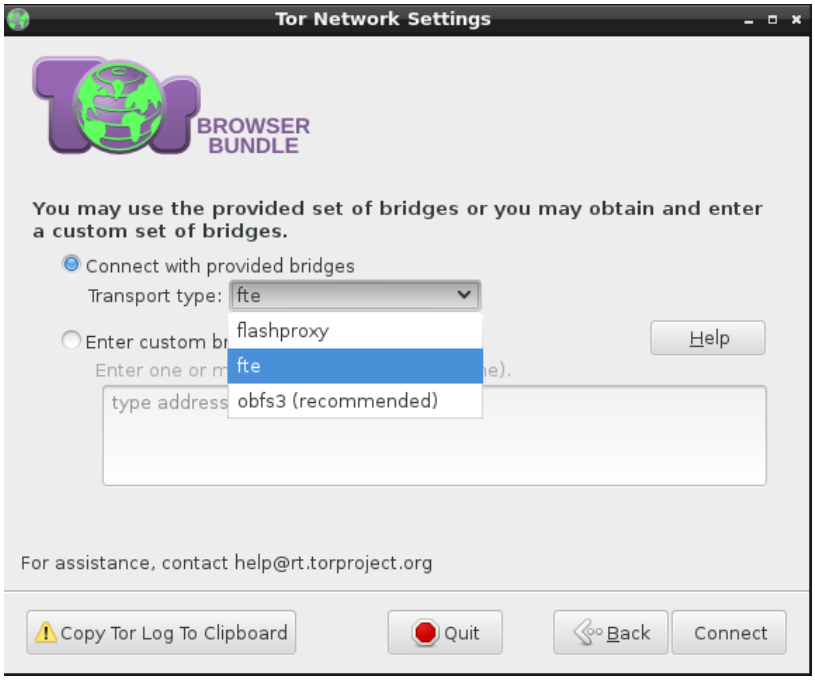
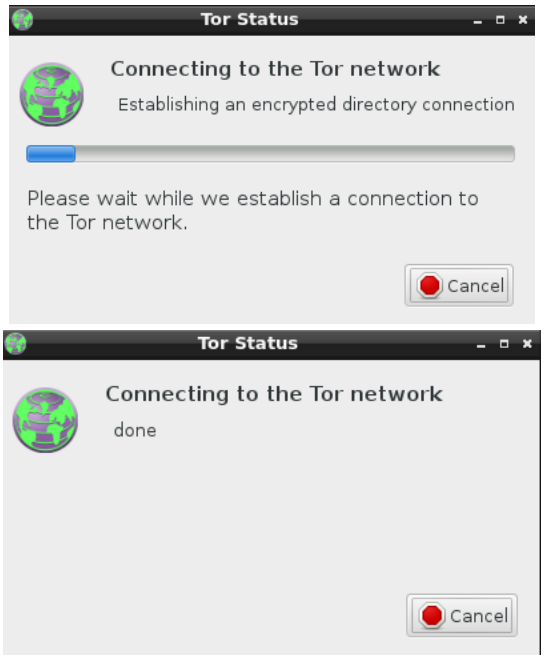


Figura 25: Configuración de bridges estándar

Los resultados volvieron a ser los mismos, por lo que no pudimos saltar la seguridad que nos proporcionó el script propuesto. A continuación podemos ver las capturas de este intento fallido.



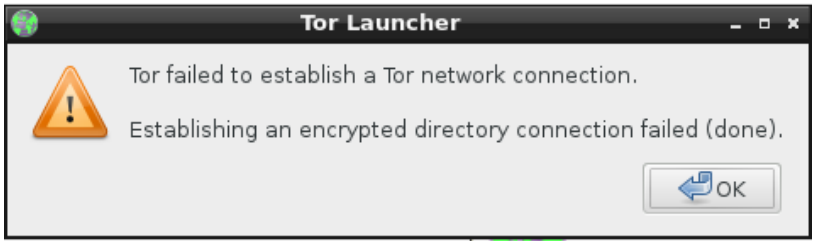


Figura 26: Nueva prueba de imposibilidad de conexión a Tor desde LAN

12. MEJORAS, AUTOMATIZACIÓN, PRUEBAS Y ANÁLISIS DE NODOS TOR

13.1. Mejoras y explicación del script iptables_tor_black_list.sh

Se hicieron algunas modificaciones y mejoras al script **iptables_tor_black_list.sh**. A continuación se describe el script completo:

```
#!/bin/bash
#

total=0
nuevos=0

# Creacion de el directorio contenedor Tor (si no existe)
mkdir -p Tor

# Descarga de la lista actual de nodos y almacenamiento en el archivo Tor_ip_list_ALL
wget http://torstatus.blutmagie.de/ip_list_all.php/Tor_ip_list_ALL.csv -O Tor/Tor_ip_list_ALL

# Ordenamiento de la lista del archivo Tor_ip_list_ALL
sort -t . -k 1,1n -k 2,2n -k 3,3n -k 4,4n Tor/Tor_ip_list_ALL -o Tor/Tor_ip_list_ALL

# Creacion de archivo Tor_black_list (si no existe)
touch Tor/Tor_black_list

# Ordenamiento de la lista del archivo Tor_black_list
sort -t . -k 1,1n -k 2,2n -k 3,3n -k 4,4n Tor/Tor_black_list -o Tor/Tor_black_list

# Posibles direcciones repetidas eliminadas
cp Tor/Tor_ip_list_ALL Tor/aux
cat Tor/aux | uniq > Tor/Tor_ip_list_ALL
rm -f Tor/aux

# Almacenamiento de las nuevas IP en el archivo Tor_black_list
# y carga de nuevas reglas iptables
for nodo in $(diff Tor/Tor_black_list Tor/Tor_ip_list_ALL | grep -E '> ([0-9]{1,3}\.){3}[0-9]{1,3}' | sed 's/^> *//g')
do
        echo $nodo >> Tor/Tor_black_list
        iptables -A FORWARD -s $nodo -j DROP
        ((nuevos++))
# Posibles direcciones repetidas eliminadas
cp Tor/Tor_black_list Tor/aux
cat Tor/aux | uniq > Tor/Tor_black_list
rm -r Tor/aux

# Revision de la black list
for nodo in $(cat Tor/Tor_black_list)
do
        iptables -C FORWARD -s $nodo -j DROP 2>/dev/null
        if [ $? -ne 0 ]; then
                iptables -A FORWARD -s $nodo -j DROP
                ((nuevos++))
        fi
        ((total++))
done

echo Nuevos nodos bloqueados: $nuevos
echo Total nodos bloqueados: $total

# Copia para analisis de metricas de Tor_ip_list_ALL
mkdir -p Tor/analisis
cp Tor/Tor_ip_list_ALL Tor/analisis/Tor_ip_list_ALL_$(date +%y%m%d%H%M)
```

Figura 27: Mejoras script iptables_tor_black_list.sh

1. Creamos el directorio contenedor **Tor** si es que no existe.

2. Descargamos la lista completa de nodos Tor y lo guardamos con el nombre de **Tor_ip_list_ALL**.

3. Ordenamos de menor a mayor las direcciones IP de **Tor_ip_list_ALL**.

4. Creamos, si no existe, el archivo **Tor_black_list** que funcionará como base de datos de las direcciones a bloquear, es decir, albergará los nodos que están activos o que alguna vez estuvieron activos dentro de la red TOR.

5. Ordenamos de menor a mayor las direcciones IP de **Tor_black_list**.

6. Se eliminan las direcciones repetidas de **Tor_black_list**. Hacemos una copia de la lista principal en el archivo temporal **aux**, ejecutamos el comando cat sobre el mismo y su salida se la entregamos al comando uniq, que volcará una sola ocurrencia de cada dirección, almacenada aux, en Tor_black_list, es decir, sobrescribe la lista principal con una lista sin direcciones repetidas. Por último, eliminamos el archivo temporal aux.

7. En el primer bucle, incorporamos a nuestra base de datos **Tor_black_list** los nodos activos que no los tengamos almacenados y cargamos la regla iptables correspondiente a cada uno de ellos.

8. Ordenamos nuevamente de menor a mayor las direcciones IP de **Tor_black_list**, ya que los nuevos nodos fueron añadidos al final del archivo.

9. Eliminamos posibles direcciones repetidas de **Tor_black_list** que pueden deberse a alguna modificación externa.

10. Recorremos **Tor_black_list**, chequeamos que para cada dirección esté cargada su correspondiente regla iptables. Si no está presente dicha regla, se añade con el comando iptables correspondiente. La salida del comando iptables que realiza el chequeo de la regla se vuelca al archivo /dev/null para eliminar la salida stderr (descriptor 2). La comprobación se realiza sobre la variable '**?**', que tendrá valor 0 si la regla está cargada y 1 si es el caso contrario.

11. Se anuncian los resultados. **Nuevos nodos bloqueados** se refiere a los nodos que están activos pero no están en Tor_black_list y los nodos que por alguna razón están en Tor_black_list pero no se encuentra cargada su regla iptables correspondiente. **Total nodos bloqueados** se refiere a la cantidad total de nodos que se encuentran en Tor_black_list y para cada uno de los cuales está cargada su correspondiente regla iptables.

12. Se crea, si no existe, el directorio analisis donde se almacenarán las listas de nodos activos para cada momento determinado en que se ejecute el script y se almacena la lista actual.

13. Se crea, si no existe, el directorio backup_tor donde se almacenarán las listas de la base de datos en cada momento determinado en que se ejecute el script y se almacena la lista actual.

De esta forma, con el mismo script, podremos cargar las reglas iptables en el momento 0, es decir, cuando no hay ninguna regla cargada. A continuación vemos un ejemplo de esto:

```
[root@192 ~]# ./iptables_tor_black_list.sh
--2014-09-02 10:37:02--  http://torstatus.blutmagie.de/ip_list_all.php/Tor_ip_list_ALL.csv
Resolviendo torstatus.blutmagie.de (torstatus.blutmagie.de)... 192.251.226.204, 2a02:cbf0:10:101::1:8324
Conectando con torstatus.blutmagie.de (torstatus.blutmagie.de)[192.251.226.204]:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: no especificado [application/force-download]
Grabando a: “Tor_ip_list_ALL”

    [    <=>                                                                 ] 86.581      69,1KB/s    en 1,2s

2014-09-02 10:37:06 (69,1 KB/s) - “Tor_ip_list_ALL” guardado [86581]

Nuevos nodos bloqueados: 8750
Total nodos bloqueados: 8750
```

Figura 28: Ejecución del script iptables_tor_black_list.sh en el momento 0

Por otro lado, los directorios analsis y backup_tor nos brindarán información que analizaremos más adelante.

13.2. Automatización del script

En primer lugar haremos que se ejecuten los scripts **iptables_tor_black_list.sh** e **iptables_squid.sh** al iniciar el sistema operativo. Para ello, crearemos el archivo **/etc/init.d/inicio.sh** que ejecutará, al iniciar el sistema operativo, los scripts que necesitemos.

```
#!/bin/bash
#

cd /root
./iptables_squid.sh
./iptables_tor_black_list.sh
exit 0
```

Figura 29: Script inicio.sh

Nota: recordamos el código del script **iptables_squid.sh**.

```
#!/bin/bash
#

iptables -F
iptables -X
iptables -F -t nat
iptables -X -t nat
iptables -t nat -A POSTROUTING -s 192.168.2.0/24 -o p2p1 -j SNAT --to 192.168.1.150
iptables -t nat -A PREROUTING -i p7p1 -p tcp --dport 80 -j DNAT --to 192.168.1.150:3128
iptables -t nat -A PREROUTING -i p2p1 -p tcp --dport 80 -j REDIRECT --to-port 3128
```

Figura 30: Script iptables_squid.sh

Luego ejecutamos el comando **chmod 755 /etc/initd.d/inicio.sh** para otorgarle al script los permisos necesarios.

```
-rwxr-xr-x  1 root root    80 sep 29 17:38 inicio.sh
```

Figura 31: Permisos script inicio.sh

Una vez realizados estos pasos, debemos crear los links simbólicos **/etc/rc3.d/S11inicio.sh**, **/etc/rc4.d/S11inicio.sh** y **/etc/rc5.d/S11inicio.sh** para que nuestro script **inicio.sh** se ejecute al iniciar en el runlevel 3, 4 y 5 respectivamente. Para ello ejecutamos los comandos:

 # ln –s /etc/init.d/inicio.sh /etc/rc3.d/S11inicio.sh

 # ln –s /etc/init.d/inicio.sh /etc/rc4.d/S11inicio.sh.

 # ln –s /etc/init.d/inicio.sh /etc/rc5.d/S11inicio.sh.

Verificamos que, en el directorio /etc/rc3.d, /etc/rc4.d y /etc/rc5.d, exista el siguiente link:

```
lrwxrwxrwx  1 root root 21 sep 29 17:51 S11inicio.sh -> /etc/init.d/inicio.sh
```

Figura 32: Link S11inicio.sh

Por otro lado utilizaremos Cron para automatizar la ejecución del script **iptables_tor_black_list.sh**. Por tal motivo, modificamos el archivo **/etc/crontab** para que se ejecute el script en los minutos 0 y 30 de cada hora del día.

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed

00,30 * * * * root /root/iptables_tor_black_list.sh
```

Figura 33: Configuración inicial crontab

13.3. Análisis de datos recolectados

Se dejó corriendo el script **iptables_tor_black_list.sh** durante 5 días y de los datos recolectados podemos ver los siguientes comportamientos de los nodos TOR.

1. Al analizar los archivos almacenados en **backup_tor**, se puede apreciar un crecimiento de nuestra base de datos. Como vemos en el siguiente gráfico, se inició la toma de datos con aproximadamente 6000 nodos y con el paso del tiempo se registraron otros 6000 nuevos nodos aproximadamente.

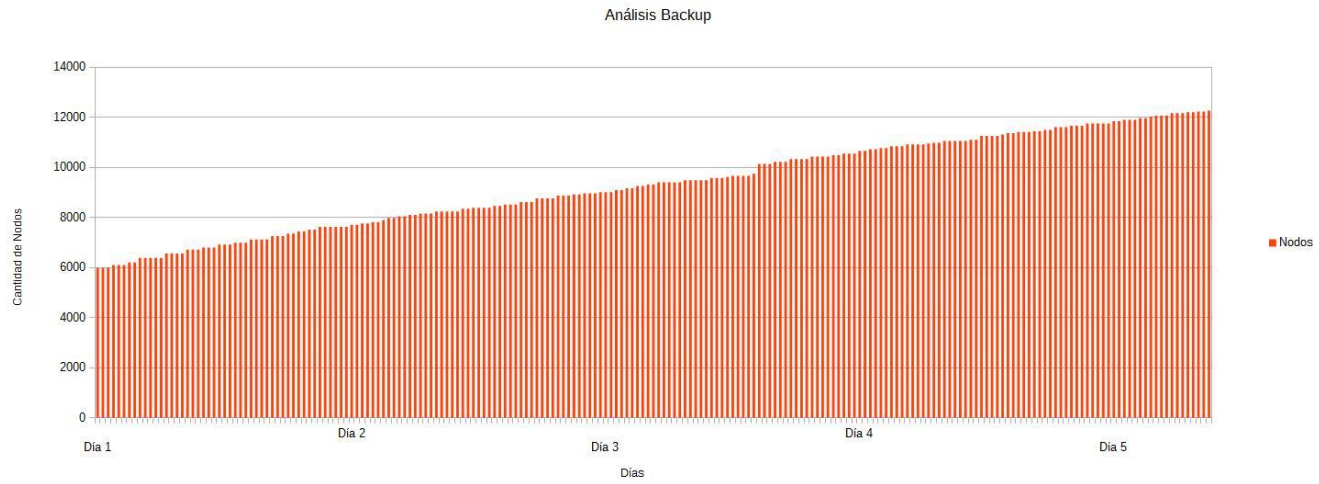


Figura 34: Gráfico Análisis Backup

2. Analizamos los archivos almacenados en **analisis**, es decir las listas de los nodos activos en el momento en que se ejecutó el script, y podemos observar como varían las cantidades de nodos activos de la red TOR a lo largo del tiempo, en promedio hay 6000 nodos activos en la red. Esto nos da un indicio de que no podemos configurar una vez nuestro firewall y no volver a hacerlo ya

que la variación de los nodos es muy amplia. Por estas razones necesitamos una configuración dinámica del firewall de nuestra infraestructura.

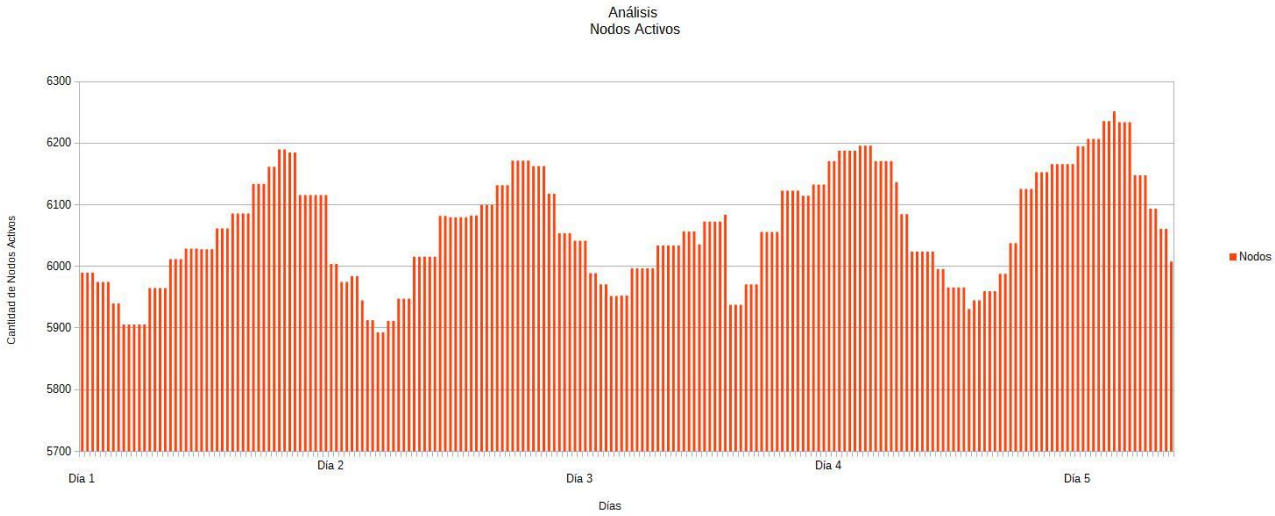


Figura 35: Gráfico Análisis Nodos Activos

3. Por último mostramos la variación de los nodos activos cada media hora durante un día (Día 1).

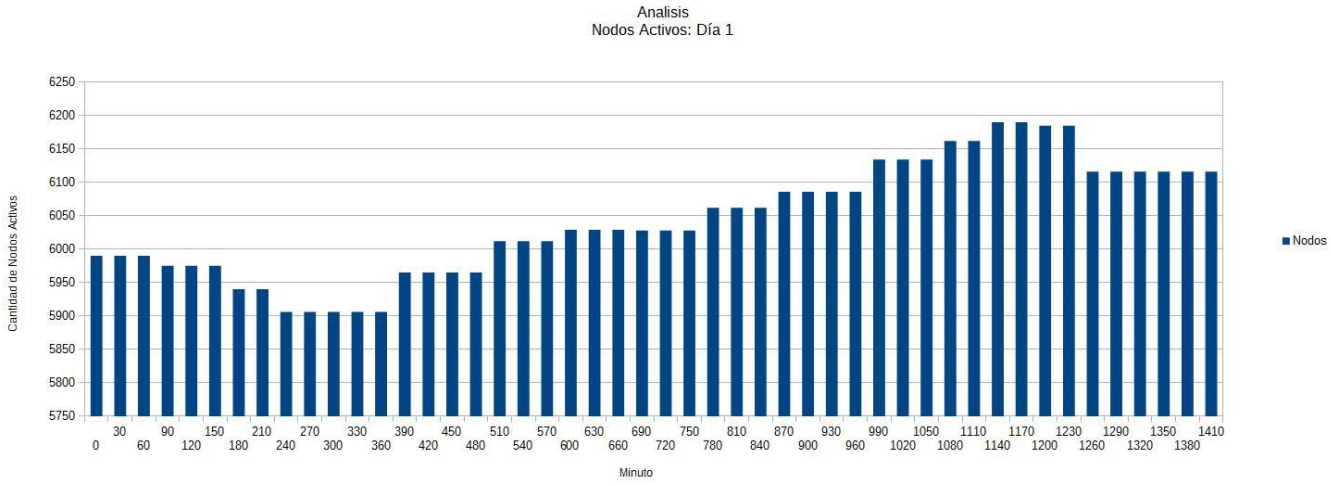


Figura 36: Gráfico Análisis Nodos Activos: Día 1

13.4. Bridges brindados por Tor

Tor nos brinda la posibilidad de conectarnos a través de sus bridges privados para evadir medidas que se hayan tomado para impedir la conexión a la red anónima. Para obtener estos bridges debemos ingresar

a la página https://bridges.torproject.org/bridges, colocar el código de seguridad (captcha) que se muestra en la misma página y obtenemos los bridges que se muestran a continuación.

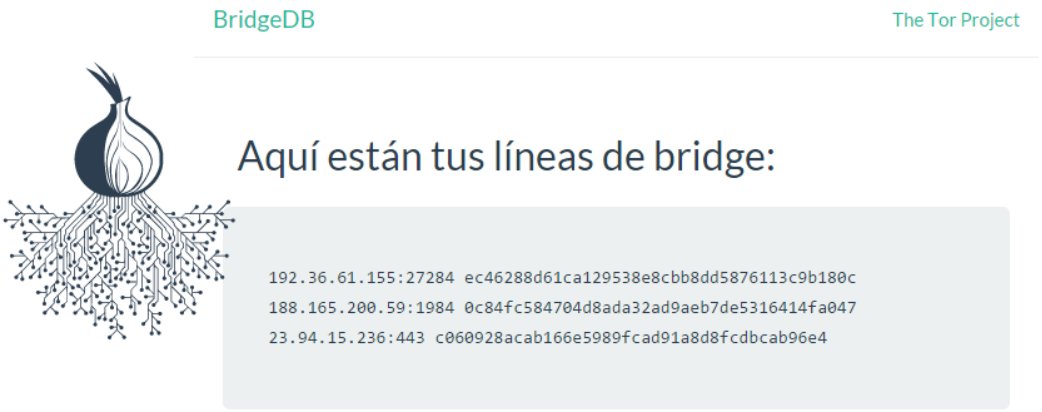


Figura 37: Bridges Tor a través de la web

Luego ingresamos a **Tor Browser**, seleccionamos la opción **Configure** (Configurar) y en la tercera pregunta, que dice que nuestro ISP está bloqueando las conexiones a la red Tor, seleccionamos la opción **Yes**:

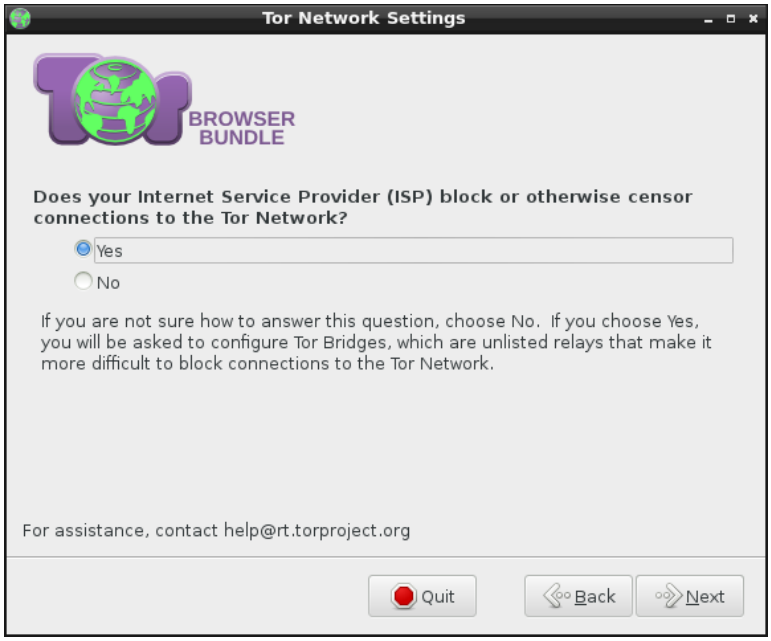


Figura 38: Nueva configuración bridges Tor 1

Seleccionamos la opción Custom Bridges, ingresamos los bridges obtenidos de la página y presionamos Connect.

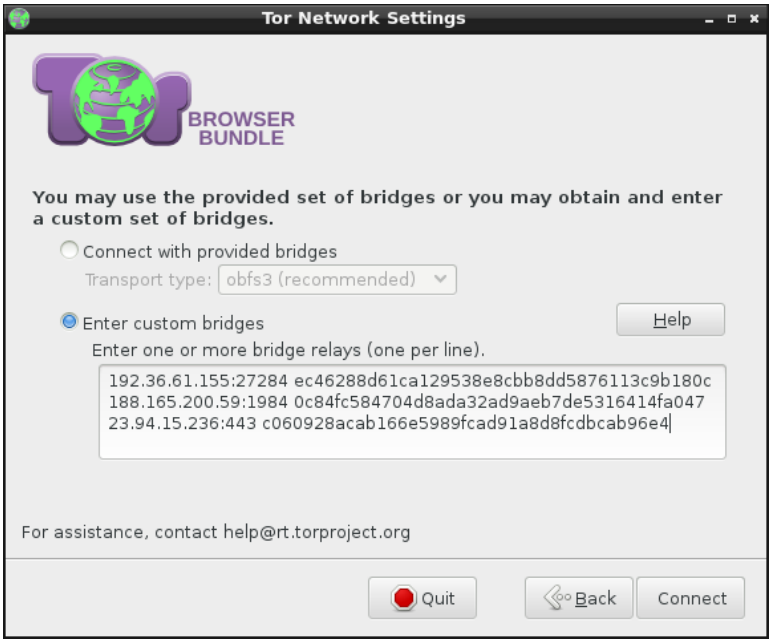


Figura 39: Nueva configuración bridges Tor 2

Como resultado podemos ver que evitamos las reglas de bloqueo que pusimos en el firewall y podemos conectarnos exitosamente, por lo que se tomarán nuevas medidas para intentar evitar este problema.

13. MEDIDA DE MITIGACIÓN CONTRA BRIDGES TOR

14.1. Introducción y explicación de medidas tomadas hasta el momento

En primera instancia y como se muestra a continuación, sin cargar las reglas del firewall iptables que bloquean los nodos Tor, podemos conectarnos libremente a la red Tor. Esto quiere decir que está permitido el tráfico desde nuestra red hacia Tor y viceversa.

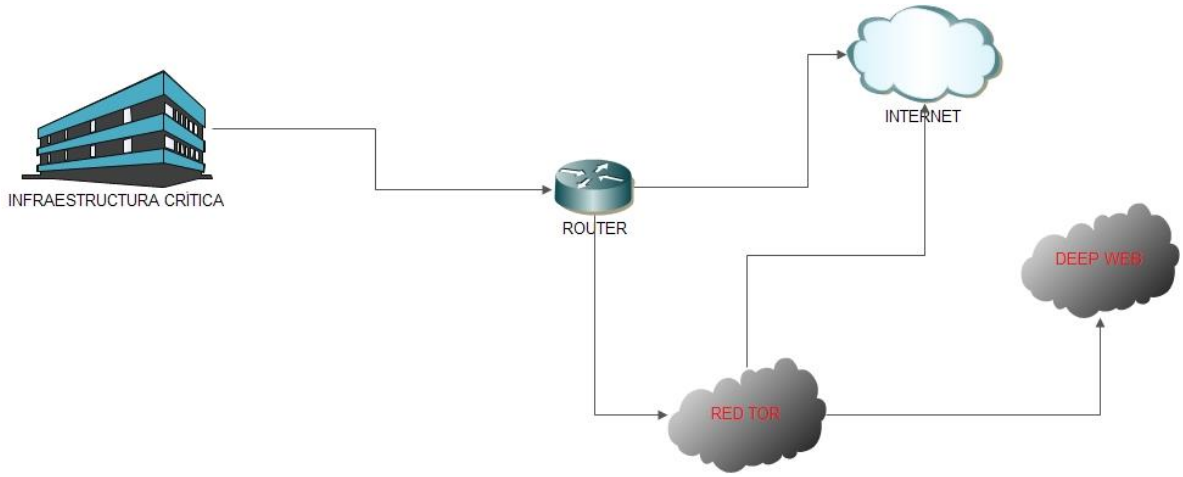


Figura 40: Esquema Infraestructura Crítica sin protección

Luego, al cargar dichas reglas en el firewall, ya no está permitido el tráfico con los nodos Tor. A continuación representamos este caso.

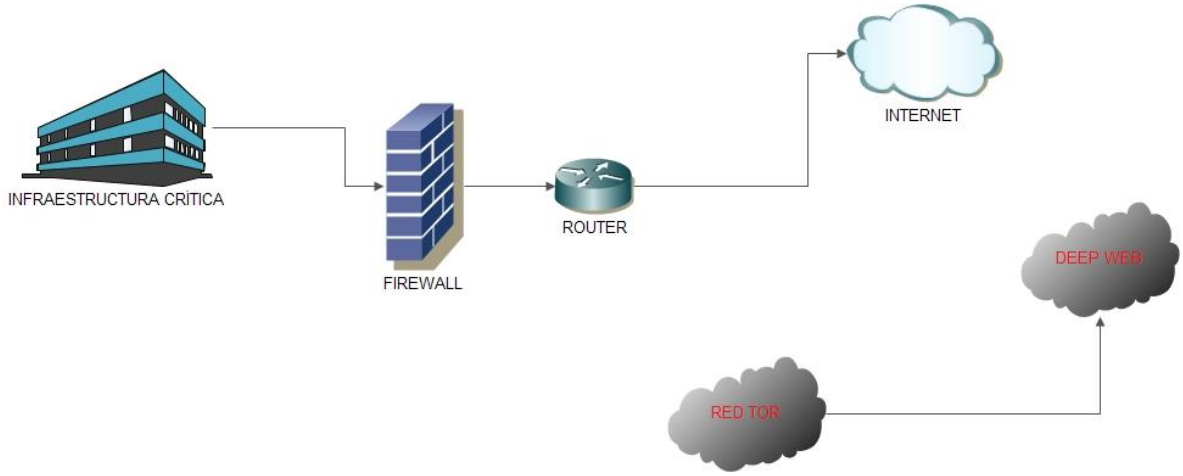


Figura 41: Esquema Infraestructura Crítica con router firewall

Como podemos ver en la siguiente imagen, Tor nos ofrece una alternativa cuando esto sucede y es utilizar Bridges, estos son nodos Tor que no están incluidos en el listado público de nodos y por lo tanto no se bloqueó el tráfico desde y hacia los mismos. De esta forma nos podemos conectar a estos bridges y ellos nos conectaran con los nodos Tor, entonces, por lo que respecta al router, solo puede ver tráfico con los bridges y no con los nodos que hemos bloqueado.

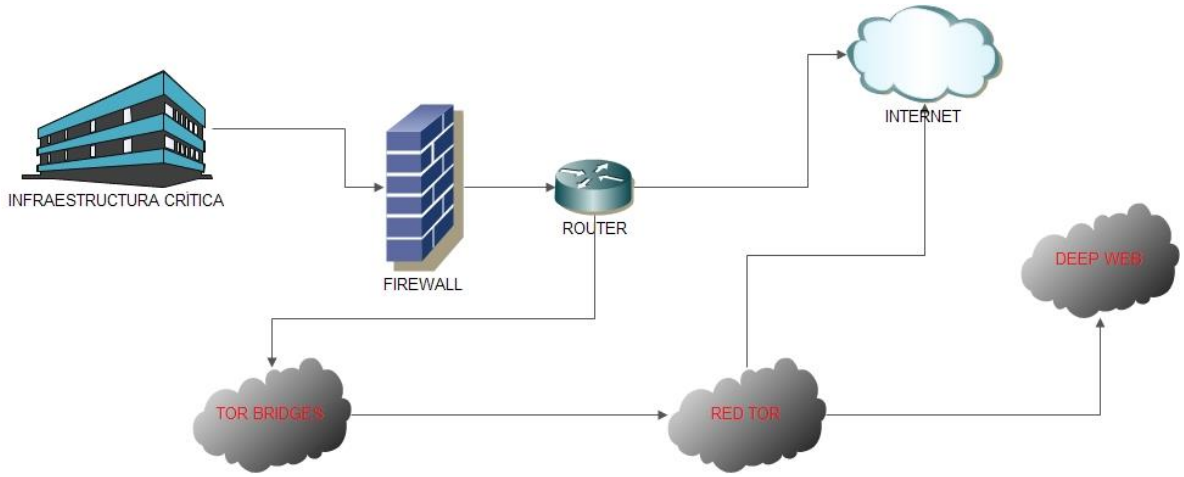


Figura 42: Esquema Infraestructura Crítica sin protección contra bridges Tor

El objetivo que se plantea en este informe es bloquear la mayor cantidad posible de bridges y de esta forma llegar al siguiente resultado.

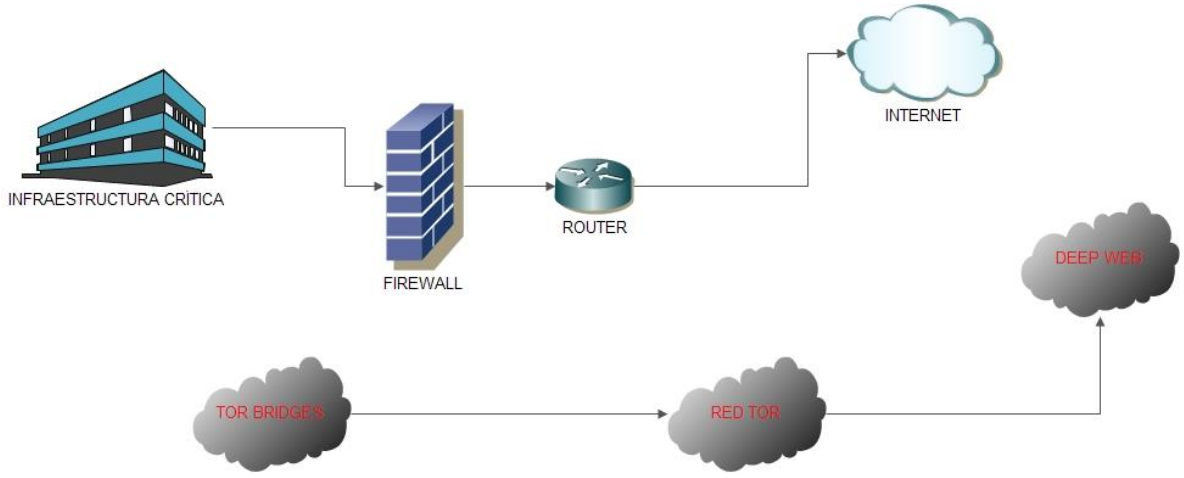


Figura 43: Esquema Infraestructura Crítica con protección contra bridges Tor

Los bridges que otorga Tor son privados y los podemos obtener por la página https://bridges.torproject.org/bridges o enviando un correo electrónico con el comando **get bridges** a la dirección bridges@torproject.org. La opción de obtener los bridges a través de la página web la descartamos ya que presenta el problema de que se debe pasar un código captcha, por lo tanto intentaremos obtener una buena cantidad de bridges privados de Tor de la manera que se explicará en los puntos posteriores.

Para realizar estas tareas debemos tener instalados los paquetes **ssmtp**, para enviar los correos y **fetchmail**, para descargar los correos a través del protocolo POP3.

14.2. Instalación y configuración de ssmtp

Si no tenemos instalado el paquete ssmtp, debemos ejecutar el comando **yum –y install ssmtp**. Para su configuración debemos editar el archivo **/etc/ssmtp/ssmtp.config** e incluir los siguientes campos con la información de la cuenta desde la que deseamos enviar un correo.

UseSTARTTLS=YES
mailhub=smtp.gmail.com:587
RewriteDomain=gmail.com
FromLineOverride=YES
UseTLS=YES
TLS_CA_File=/etc/pki/tls//certs/ca-bundle.crt
AuthUser=usuario
AuthPass=contraseña
root=usuario@gmail.com
Hostname=usuario@gmail.com

14.3. Instalación y configuración de fetchmail

Si no tenemos instalado el paquete fetchmail, debemos ejecutar el commando **yum –y install fetchmail** y una vez instalado veremos el siguiente mensaje.

```
Instalado:
  fetchmail.i686 0:6.3.22-2.fc18
```

Figura 44: Instalación fetchmail

Para configurar fetchmail, debemos crear el archivo **.fetchmailrc** en el directorio del usuario que utilizará la cuenta (en nuestro caso es **/home/router/.fetchmailrc**) y editarlo para que contenga las siguientes líneas.

 # set username
Set postmaster “router”

Pool pop.gmail.com with proto POP3
User ‘usuario@gmail.com’ there with password ‘clave’ is router here options ssl

14.4. Script send_mail_bridges.sh

Este script se ejecutará como root ya que necesitamos permisos para modificar determinados archivos. Podemos ver el script en la siguiente imagen y su explicación debajo de la misma.

```
#!/bin/bash
#

# Enviamos los emails desde cada una de las 10 cuentas de correo
for((i=0; i<10; i++))
do
        # Carga del template en el archivo de config de ssmtp
        cat template_ssmtp > /etc/ssmtp/ssmtp.conf

        # Carga de los datos particulares de la cuenta
        usuario=prbrtor$i
        cuenta=$usuario@gmail.com
        echo -e "AuthUser="$usuario"\nAuthPass=prbrtor123456789\nroot="$cuenta"\nHostname="$cuenta >> /etc/ssmtp/ssmtp.conf

        echo -e "To: bridges@torproject.org\nFrom: "$cuenta"\nSubject: get bridges\n\nget bridges" > mail

        # Enviamos el email para obtener los bridges
        ssmtp bridges@torproject.org < mail

        # Verificamos si se envio correctamente
        if [ $? -ne 0 ]; then
                echo "E-mail_"$i", se produjo un error"
        else
                echo "E-mail_"$i", enviado correctamente"
        fi
        sleep 5
done
```

Figura 45: Script send_mail_bridges.sh

1. Realizamos un bucle para enviar un email con cada una de la cuentas. Hemos creado 10 cuentas de correo electrónico en Gmail:

    a. prbrtor0@gmail.com
    b. prbrtor1@gmail.com
    c. prbrtor2@gmail.com
    d. prbrtor3@gmail.com
    e. prbrtor4@gmail.com
    f. prbrtor5@gmail.com
    g. prbrtor6@gmail.com
    h. prbrtor7@gmail.com
    i. prbrtor8@gmail.com
    j. prbrtor9@gmail.com

    Por otro lado, al intentar acceder a cada una de estas cuentas a través de ssmtp por línea de comandos, Google nos envía un email con el siguiente contenido:

Figura 46: Confirmación de inicio de sesión por ssmtp

Hacemos click en el link que nos ofrece la opción Si has sido tú y habilitamos el acceso a nuestra cuenta desde aplicaciones menos seguras.

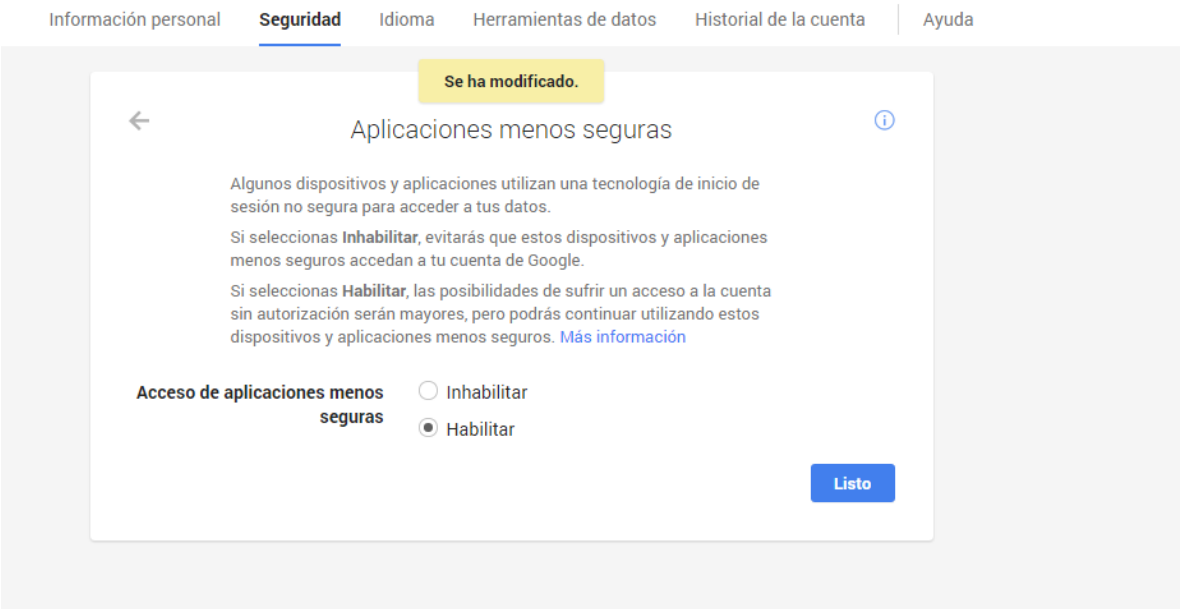


Figura 47: Habilitación de aplicaciones menos seguras en Gmail

2. Cargamos el contenido del archivo **template_ssmtp** en el archivo de configuración de ssmtp, **/etc/ssmtp/ssmtp.conf**. Este contenido es el que no se modificará para cada cuenta:

```
UseSTARTTLS=YES
mailhub=smtp.gmail.com:587
RewriteDomain=gmail.com
FromLineOverride=YES
UseTLS=YES
TLS_CA_File=/etc/pki/tls/certs/ca-bundle.crt
```

Figura 48: Configuración ssmtp

3. Agregamos al archivo de configuración de ssmtp los datos particulares de cada cuenta.
4. Escribimos los datos del email a enviar en el archivo mail. Por ejemplo:

```
To: bridges@torproject.org
From: prbrtor6@gmail.com
Subject: get bridges

get bridges
```

Figura 49: Ejemplo de correo a enviar solicitando bridges Tor

5. Enviamos el correo a través del comando ssmtp con el contenido del archivo mail.
6. Informamos si se ha enviado correctamente el email de cada cuenta y esperamos 5 segundos para no tener problemas con ssmtp antes de enviar el próximo email.

14.5. Script get_bridges.sh

Este script lo ejecutará el usuario router ya que utilizaremos su archivo **/var/spool/mail/router** para almacenar temporalmente los emails que descarguemos por POP3. Podemos ver el script en la siguiente imagen y su explicación debajo de la misma.

```
#!/bin/bash
#

# En caso de no existir, creamos el archivo de config de fetchmail y le otorgamos los permisos necesarios
touch .fetchmailrc
chmod 700 .fetchmailrc

# Creamos, si no exite, el directorio contenedor
mkdir -p Tor_bridges

for((i=0; i<10; i++))
do
        # Carga del template
        cat template_fetchmailrc > .fetchmailrc

        # Carga de datos de la cuenta
        cuenta=prbrtor$i@gmail.com
        echo "        user '$cuenta' there with password 'prbrtor123456789' is router here options ssl" >> .fetchmailrc

        # Borramos los mails de router
        > /var/spool/mail/router

        # Descargamos los nuevos mails de la cuenta
        fetchmail -d0 -vk -s pop.gmail.com

        # Extraemos los 3 bridges que nos otorga Tor y los guardamos en blacklist_bridges
        touch Tor_bridges/blacklist_bridges
        linea=`cat -n /var/spool/mail/router | grep "Here are your bridges:" | grep -oiE '([0-9]){2}'`
        sed -n $((linea+=2)),$((linea+=2))p /var/spool/mail/router | grep -oiE '([0-9]{1,3}\.){3}[0-9]{1,3}' >> Tor_bridges/bl
acklist_bridges

        # Impirmimos informacion
        echo "Descarga "$i" finalizada"
done

# Ordenamos de menor a mayor el archivo blacklist_bridges
sort -t . -k 1,1n -k 2,2n -k 3,3n -k 4,4n Tor_bridges/blacklist_bridges -o Tor_bridges/blacklist_bridges

# Posibles direcciones repetidas eliminadas
cp Tor_bridges/blacklist_bridges Tor_bridges/aux
cat Tor_bridges/aux | uniq > Tor_bridges/blacklist_bridges
rm -f Tor_bridges/aux

# Copia de seguridad de blacklist_bridges
mkdir -p Tor_bridges/backup_bridges
cp Tor_bridges/blacklist_bridges Tor_bridges/backup_bridges/blacklist_bridges_$(date +%y%m%d%H%M)
```

Figura 50: Script get_bridges.sh

1. Creamos el archivo **.fetchmail** si es que no existe y le otorgamos los permisos correspondientes.
2. Si no existe creamos el directorio **Tor_bridges**.
3. Hacemos un bucle para recorrer cada una de las 10 cuentas.
4. Cargamos el contenido del archivo **template_fetchmailrc** en el archivo **.fetchmailrc**. Estos son los datos generales y no particulares de cada cuenta.

```
# set username
set postmaster "router"

poll pop.gmail.com with proto POP3
```

Figura 51: Archivo .fetchmailrc

5. Añadimos los datos de la cuenta actual al archivo **.fetchmailrc**.
6. Eliminamos los correo viejos de **/var/spool/mail/router**.
7. Ejecutamos el comando fetchmail para descargar los emails nuevos de la cuenta actual.
8. Luego buscamos la cadena "**Here are your bridges:**", obtenemos el número de línea en que se encuentra y sabemos que dos líneas más abajo está la primer dirección del primer bridge de los

tres. Luego, a través de expresiones regulares, obtenemos las direcciones IP que necesitamos con los comandos sed y grep y los agregamos al final de la blacklist.

9. Ordenamos las direcciones IP de la blacklist de menor a mayor.
10. Creamos el directorio **Tor_bridges/backup_bridges/** y realizamos una copia de seguridad del archivo **blacklist_bridges**.

14.6. Automatización de los scripts send_mail_bridges.sh y get_bridges.sh

Para automatizar los scripts utilizaremos Cron, por lo que debemos editar el archivo **/etc/crontab**. El **script send_mail_bridges.sh** se ejecutará en el minuto 15 cada 3 horas empezando desde las 0 horas y el script **get_bridges.sh** se ejecutará en el minuto 17 cada 3 horas empezando desde las 0 horas. Se deben ejecutar cada 3 horas ya que el servidor que nos otorga los bridges impone dicha regla.

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed

00,30 * * * * root /root/iptables_tor_black_list.sh
15 0,3,6,9,12,15,18,21 * * * root /root/send_mail_bridges.sh
17 0,3,6,9,12,15,18,21 * * * router /home/router/get_bridges.sh
```

Figura 52: Configuración crontab

14.7. Bloqueo de los bridges contenidos en blacklist_bridges

Para bloquear los bridges que están almacenados en blacklist_bridges, agregamos las siguientes líneas en el script **iptables_tor_black_list.sh** luego de la línea **touch Tor_black_list**.

```
# Agregamos los bridges privados a la lista
cat /home/router/Tor_bridges/blacklist_bridges >> Tor/Tor_black_list
```

Figura 53: Nueva línea en script iptables_tor_black_list.sh

De esta manera se adicionan los bridges al final de la lista principal y luego se realiza el proceso normal como si fuesen nodos comunes de Tor.

Luego si queremos utilizar alguno de los bridges bloqueados para ingresar a Tor podemos ver que no es posible. Se utilizaron y bloquearon los mismos bridges utilizados en la prueba en la que tuvimos éxito evitando la seguridad del firewall.

```
Oct 13 18:37:39.000 [notice] New control connection opened.
Oct 13 18:38:43.000 [warn] Problem bootstrapping. Stuck at 5%: Connecting to dir
ectory server. (Connection timed out; TIMEOUT; count 3; recommendation warn)
Oct 13 18:38:43.000 [warn] 2 connections have failed:
Oct 13 18:38:43.000 [warn]  2 connections died in state connect()ing with SSL st
ate (No SSL object)
Oct 13 18:38:43.000 [notice] Ignoring directory request, since no bridge nodes a
re available yet.
Oct 13 18:38:44.000 [notice] Delaying directory fetches: No running bridges
Oct 13 18:38:55.000 [notice] Application request when we haven't used client fun
ctionality lately. Optimistically trying known bridges again.
Oct 13 18:39:20.000 [notice] Application request when we haven't used client fun
ctionality lately. Optimistically trying directory fetches again.
Oct 13 18:40:23.000 [warn] Problem bootstrapping. Stuck at 5%: Connecting to directory server. (Connection timed out; TIMEOUT;
 count 4; recommendation warn)
Oct 13 18:40:23.000 [warn] 3 connections have failed:
Oct 13 18:40:23.000 [warn]  3 connections died in state connect()ing with SSL state (No SSL object)
Oct 13 18:40:54.000 [notice] Owning controller connection has closed -- exiting now.
```

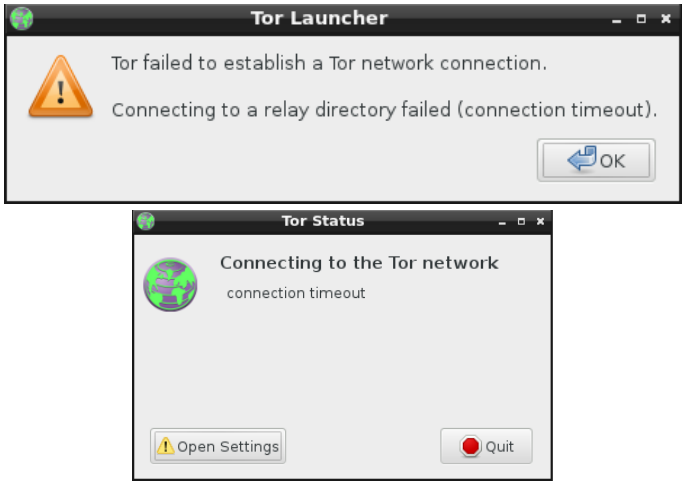


Figura 54: Evidencia de bloqueo de bridges Tor

14. GEOLOCALIZACIÓN DE NODOS TOR

15.1. Introducción

En este apartado se conformará una base de datos en MongoDB a partir de todos los nodos Tor recolectados hasta el momento y la información asociada a cada dirección IP de los mismos. Luego se ubicará cada nodo Tor en un mapa a través de sus coordenadas asociadas.

15.2. Instalación de MongoDB

MongoDB es una base de datos orientada a documentos. Esto quiere decir que en lugar de almacenar los datos en registros, almacena los datos en documentos. Estos documentos son BSON, que es una representación binaria de JSON.

Una de las diferencias más importantes con respecto a las bases de datos relacionales, es que no es necesario seguir un esquema. Los documentos de una misma colección (concepto similar a una tabla de una base de datos relacional) pueden tener esquemas diferentes.

Para instalar MongoDB ejecutamos el siguiente comando:

 # yum install mongodb-server mongodb

Si tenemos éxito veremos un mensaje similar al siguiente:

```
Instalado:
  mongodb.i686 0:2.4.6-1.fc18          mongodb-server.i686 0:2.4.6-1.fc18

Dependencia(s) instalada(s):
  boost-chrono.i686 0:1.50.0-7.fc18
  boost-filesystem.i686 0:1.50.0-7.fc18
  boost-iostreams.i686 0:1.50.0-7.fc18
  boost-program-options.i686 0:1.50.0-7.fc18
  boost-system.i686 0:1.50.0-7.fc18
  boost-thread.i686 0:1.50.0-7.fc18
  snappy.i686 0:1.0.5-2.fc18
  v8.i686 1:3.14.5.10-3.fc18

¡Listo!
```

Figura 55: Instalación MongoDB

Lo primero que debemos hacer es ejecutar el siguiente comando para poner a funcionar el servicio de mongo:

 # systemctl start mongod

Si al ejecutar el comando **mongo** obtenemos el siguiente error, debemos verificar que el servicio esté funcionando correctamente:

```
[root@192 ~]# mongo
MongoDB shell version: 2.4.6
connecting to: test
Sun Dec  7 19:03:47.192 Error: couldn't connect to server 127.0.0.1:27017 at src
/mongo/shell/mongo.js:145
exception: connect failed
```

Figura 56: Error de conexión MongoDB

Si todo funciona bien, al ejecutar el comando mongo accedemos a la terminal de mongodb:

```
[root@192 ~]# mongo
MongoDB shell version: 2.4.6
connecting to: test
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
Mon Dec  8 11:06:20.228 [initandlisten]
Mon Dec  8 11:06:20.228 [initandlisten] ** NOTE: This is a 32 bit MongoDB binary
.
Mon Dec  8 11:06:20.228 [initandlisten] **       32 bit builds are limited to le
ss than 2GB of data (or less with --journal).
Mon Dec  8 11:06:20.228 [initandlisten] **       See http://dochub.mongodb.org/c
ore/32bit
Mon Dec  8 11:06:20.228 [initandlisten]
> 
```

Figura 57: Terminal de MongoDB

15.3. Generación de BD

Primero vamos a crear la base de datos TOR e indicaremos que trabajamos con ella. Esto se realiza con el comando **use** dentro de la terminal de mongodb:

```
> use TOR
switched to db TOR
```

Figura 58: Creación de base de datos TOR en MongoDB

Para generar una colección e insertar objetos en ella utilizaremos el comando insert. Por ejemplo:

```
> db.prueba.insert({"ejemplo": "esto es una prueba"})
```

Figura 59: Ejemplo de creación de una colección con objectos

Para consultar el contenido de una colecciona utilizamos el comando find. Por ejemplo, para ver todo el contenido de prueba:

```
> db.prueba.find()
{ "_id" : ObjectId("5485bbdc9cedf50fd4140fbb"), "ejemplo" : "esto es una prueba" }
```

Figura 60: Consulta del contenido de la colección prueba

Se puede observar que por defecto se nos asigna un id de objeto. Para eliminar una colección utilizamos el comando **remove**. Por ejemplo:

>db.prueba.remove()

15.4. Script mongodb_tor.sh para poblar la Base de Datos

Vamos a utilizar el comando **curl ipinfo.io/<ip>** para consultar la información de cada una de las IP de los nodos Tor recolectados hasta ahora. Este comando realiza una consulta a una base de datos online a partir de la IP dada como parámetro y nos devuelve información sobre esa IP en formato JSON. Por ejemplo para la IP 201.253.34.15 obtenemos:

```
[root@192 ~]# curl ipinfo.io/201.253.34.15
{
  "ip": "201.253.34.15",
  "hostname": "host15.201-253-34.telecom.net.ar",
  "city": null,
  "region": null,
  "country": "AR",
  "loc": "-34.6033,-58.3817",
  "org": "AS7303 Telecom Argentina S.A."
}[root@192 ~]#
```

Figura 61: Información obtenida a través del comando curl ipinfo.io

De esta forma la idea es insertar cada uno de estos objetos en una colección tor_nodes, obteniendo las IP de Tor_black_list para realizar las consultas y más adelante utilizar las coordenadas de cada nodo para armar un mapa de geolocalización de los nodos Tor.

Entonces el script **/root/mongodb/mongodb_tor.sh** se compone de la siguiente manera:

```
#!/bin/bash
#

# Contador de objetos
total=0

# Si existe la coleccion tor_nodes, la borramos
mongo TOR <<EOF
db.tor_nodes.remove();
EOF

# Poblamos la base de datos a partir de cada IP de Tor_black_list
for nodo in $(cat /root/Tor/Tor_black_list)
do

        object=`curl ipinfo.io/$nodo`
        mongo TOR <<EOF
        db.tor_nodes.insert($object);
EOF
        echo Nodo: $nodo
        ((total++))
done

# Mensaje de informacion
echo Cantidad de objetos cargados: $total
```

Figura 62: Script mogodb_tor.sh

1. Eliminamos la colección **tor_nodes** para cargar los datos desde cero.
2. Luego recorremos el archivo **Tor_black_list** para obtener las IP de cada uno de los nodos Tor.
3. Guardamos en la variable **object** el texto JSON obtenido de la base de datos online con la información perteneciente a la IP en cuestión.
4. Nos conectamos a la base de datos **TOR** e insertamos la información de **object** en la colección **tor_nodes**.
5. El script brinda información sobre la carga en mongodb, el nodo actual procesado y por último la cantidad de nodos cargados en la base de datos.

Nota: se utiliza el indicador **EOF** (End Of File) tanto para ingresar al shell de mongodb y poder ejecutar comandos desde bash como para salir de mongodb.

A continuación vemos la última parte de la salida que proporciona el script:

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   203  100   203    0     0    523      0 --:--:-- --:--:-- --:--:--  1020
MongoDB shell version: 2.4.6
connecting to: TOR
bye
Nodo: 223.238.122.91
Cantidad de objetos cargados: 64327
```

Figura 63: Resultado de la ejecución del script mongodb_tor.sh en el momento 0

15.5. Instalación de Google Earth

Primero debemos descargar la herramienta Google Earth de la página https://www.google.es/intl/es_es/earth/download/ge/agree.html.

Una vez descargado el paquete google-earth-stable_current_i386.rpm ejecutamos el siguiente comando para instalarlo.

yum install google-earth-stable_current_i386.rpm

Si obtenemos un error debemos instalar la herramienta rpmrebuild:

yum install rpmrebuild

Si tenemos éxito veremos el siguiente mensaje informativo:

```
Instalado:
  rpmrebuild.noarch 0:2.9-1.fc18

Dependencia(s) instalada(s):
  patch.i686 0:2.6.1-14.fc18          rpm-build.i686 0:4.10.3.1-3.fc18

¡Listo!
```

Figura 64: Instalación de rpmrebuild

Ejecutamos el siguiente comando para modificar el paquete RPM de Google:

 # rpmrebuild -ep google-earth-stable_current_x86_64.rpm

Donde la opcion **e** correponde a editar y la opción **p** a paquete.

Buscamos y comentamos o eliminamos eliminamos la siguiente línea:

%dir %attr(0755, root, root) "/usr/bin"

Esto generará el paquete RPM **/root/rpmbuild/RPMS/x86_64/google-earth-stable-algo.x86_64.rpm**. Para instalarlo ejecutamos el siguiente comando:

 # yum localinstall /root/rpmbuild/RPMS/x86_64/google-earth-stable-algo.x86_64.rpm

Si tenemos éxito veremos un mensaje como el siguiente:

```
Instalado:
  google-earth-stable.i386 0:7.1.2.2041-0

Dependencia(s) instalada(s):
  libpng12.i686 0:1.2.50-2.fc18
  pax.i686 0:3.4-14.fc18
  qt3.i686 0:3.3.8b-43.fc18
  redhat-lsb-core.i686 0:4.1-10.fc18
  redhat-lsb-desktop.i686 0:4.1-10.fc18
  redhat-lsb-submod-multimedia.i686 0:4.1-10.fc18
  redhat-lsb-submod-security.i686 0:4.1-10.fc18

¡Listo!
```

Figura 65: Instalación de Google Earth

15.6. Geolocalización de los nodos

Lo primero que debemos hacer es exportar la colección **tor_nodes** a un archivo CSV con el comando **mongoexport**:

```
[root@192 /]# mongoexport --csv -d TOR -c tor_nodes -f "ip","hostname","city","r
egion","country","loc","org","postal" -o tor_nodes.csv
```

Figura 66: Comando de exportación de base de datos TOR a archivo CSV

Dónde:

--csv: es el formato de archivo de salida.

-d: es la base de datos con la que trabajamos.

-c: es la colección que vamos a exportar.

-f: son los campos que incluiremos en el archivo CSV.

-o: es el nombre del archivo de salida.

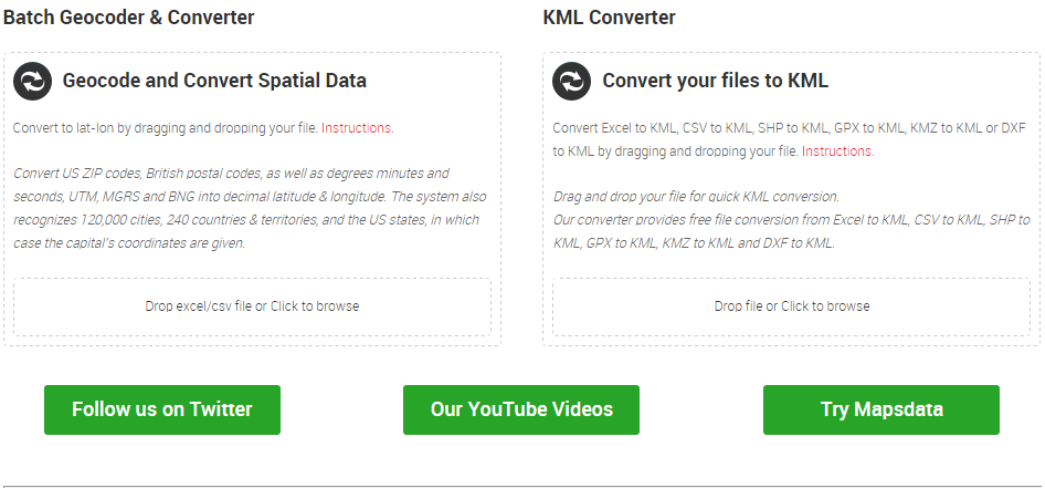
Luego debemos separar la columna **"loc"** en **Latitude** y **Longitude** y reorganizar el archivo en el siguiente orden **Latitude, Longitude, Name, Description, Icon:**

	A	B	C	D	E
1	Latitude	Longitude	Name	Description	Icon
2	7.9090	98.3332	AS9737 TOT Public Company Limited	1.0.243.83	
3	2.7867	101.9953	AS4788 TM Net, Internet Service Provider	1.9.180.118	
4	28.6000	77.2000	AS45528 Tikona Digital Networks Pvt Ltd.	1.22.128.211	

Figura 67: Formato del archivo CSV

El campo **org** pasa a ser **Name** y el campo **ip** pasa a ser **Description**. Esto se debe a que la herramienta Google Earth utiliza esta convención.

Vamos a convertir el archivo CSV en KML a traves de la siguiente página http://www.mapsdata.co.uk/online-file-converter/ . Vamos a arrastrar el nuestro archivo CSV al lugar indicado en la página web, se cargará y luego bajaremos el archivo KML.



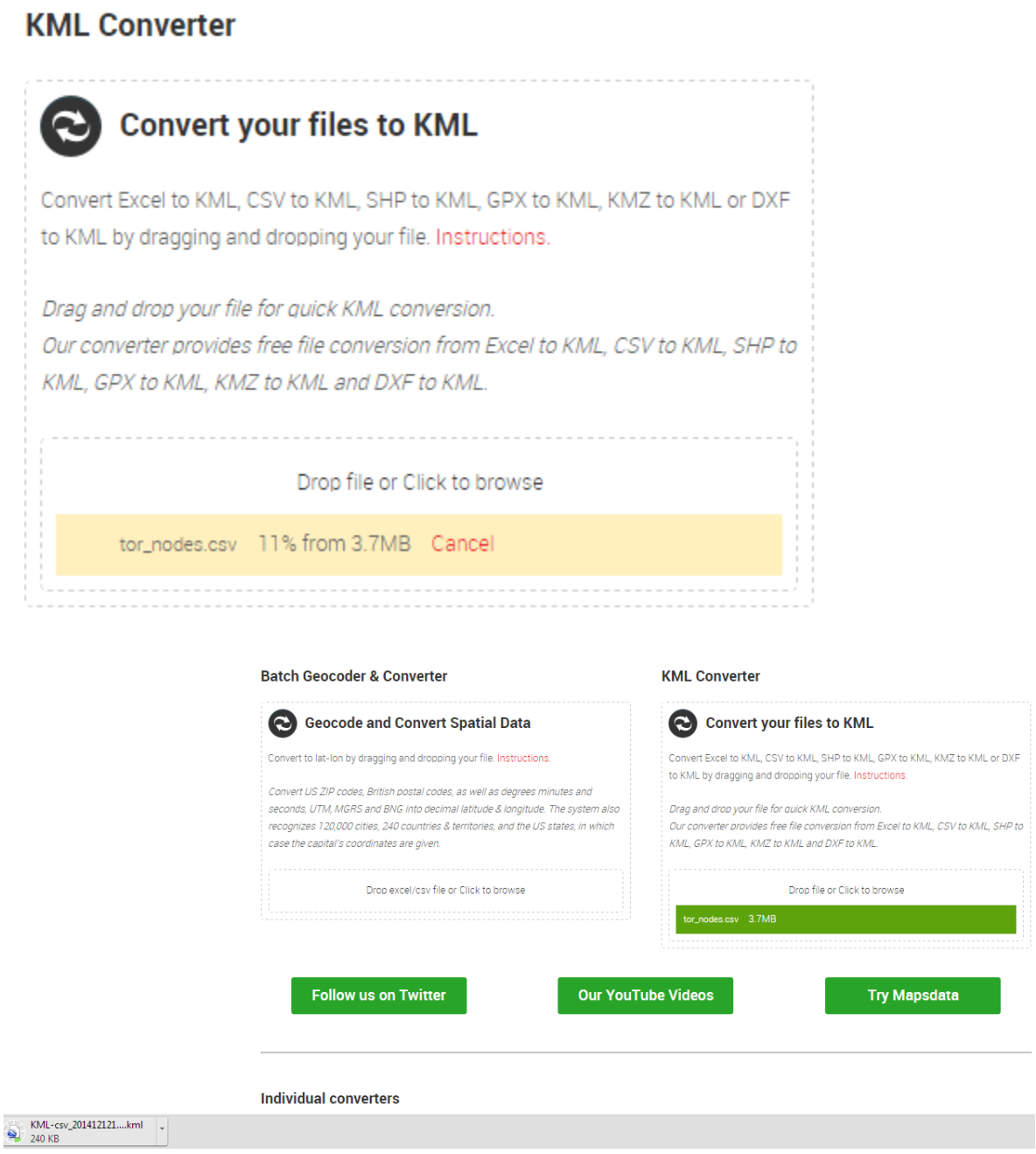


Figura 68: Pasos para convertir archivo CSV a KML

15.6.1. Google Earth

Accedemos a Google Earth, vamos a **Archivo -> Abrir…** y seleccionamos el archivo **tor_nodes.kml**.

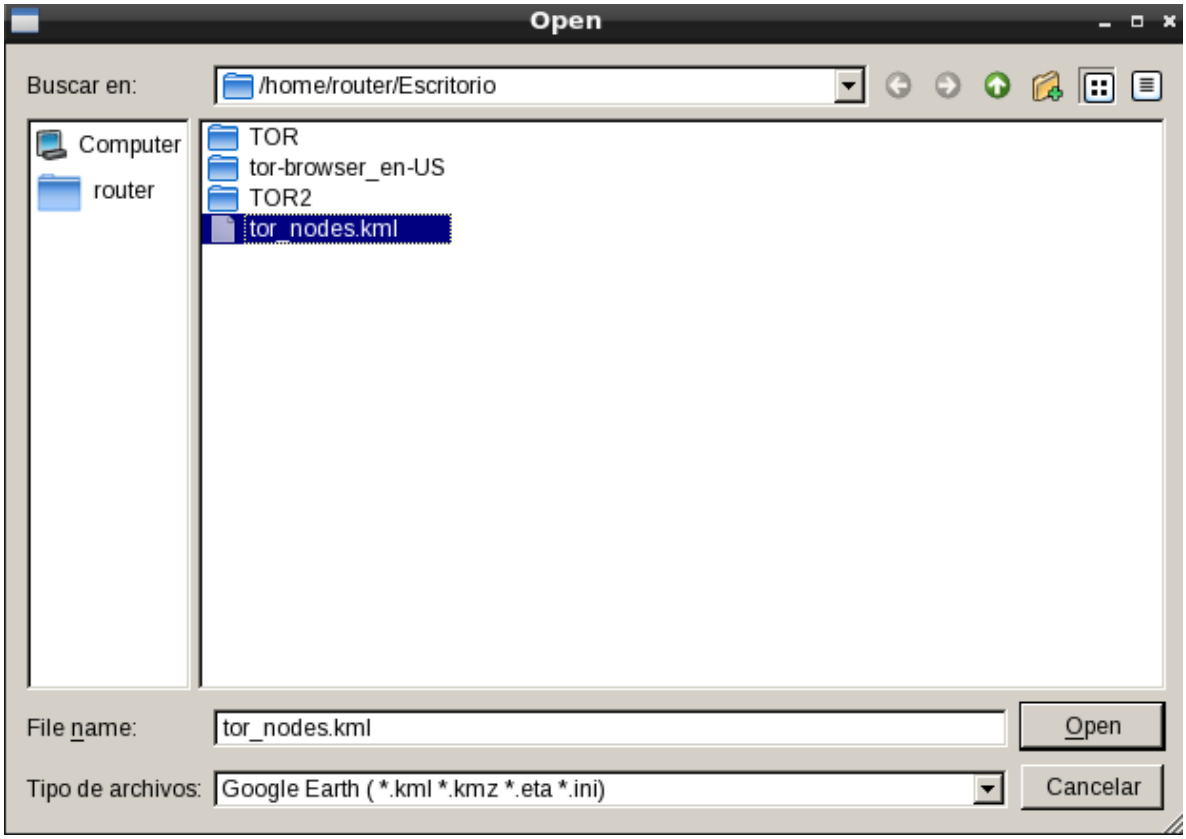


Figura 69: Carga de archivo tor_nodes.kml en Google Earth

Podemos ver en 3 dimensiones los nodos que cargamos con su correspondiente descripción:

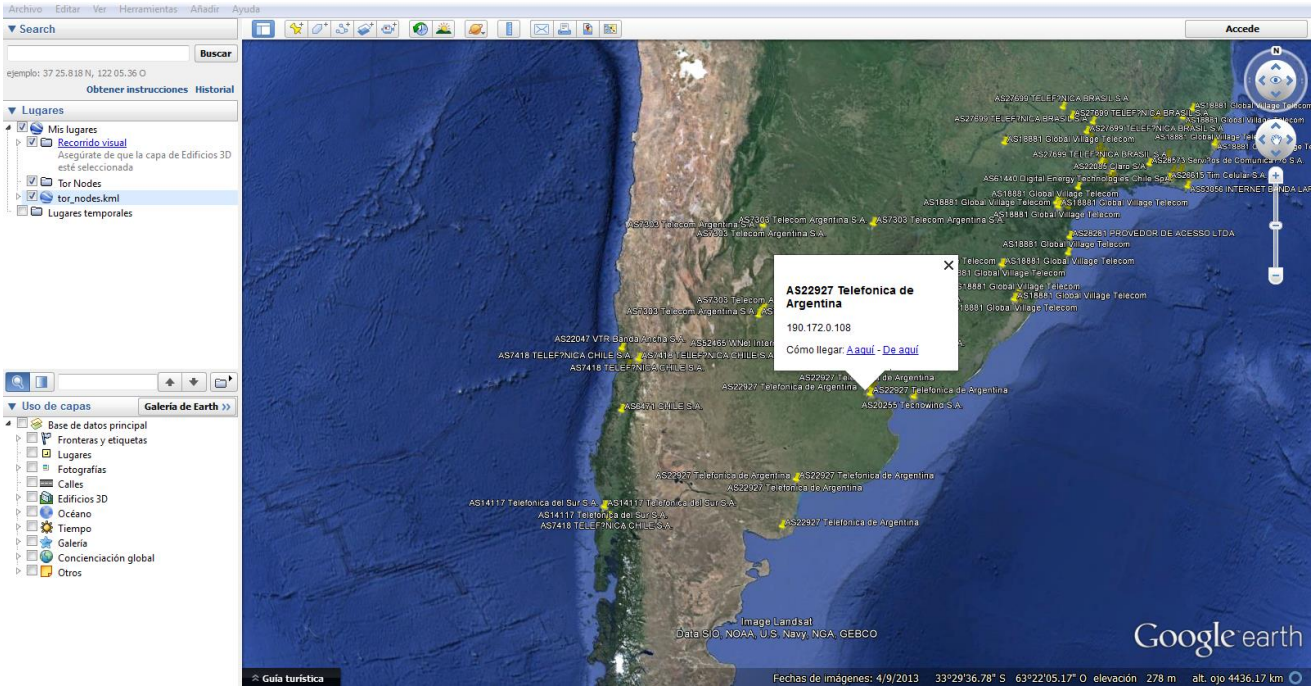


Figura 70: Vista de nodos Tor en Google Earth

El problema con Google Earth es que con tanta cantidad de nodos e información consume demasiados recursos y se hace imposible navegar el mapa.

15.6.2. ArcGIS Explorer Desktop

Por otro lado, ArcGIS Explorer Desktop es una herramienta para Microsoft Windows que nos permite visualizar la información en mapas de dos dimensiones y sin tanta carga como Google Earth. Utilizaremos dicha herramienta para cargar el archivo KML y representarlo en un mapa mundial. Para descargar el programa accedemos a http://www.esri.com/software/arcgis/explorer-desktop/download y luego lo instalamos siguiendo los pasos del Wizzard.

Una vez instalado este programa, accedemos a él y vamos a **Menu -> New** y creamos un mapa vacío. Luego vamos a **Map -> Add Content -> KML Files -> KML Files…** y cargamos el archivo KML de tor_nodes.

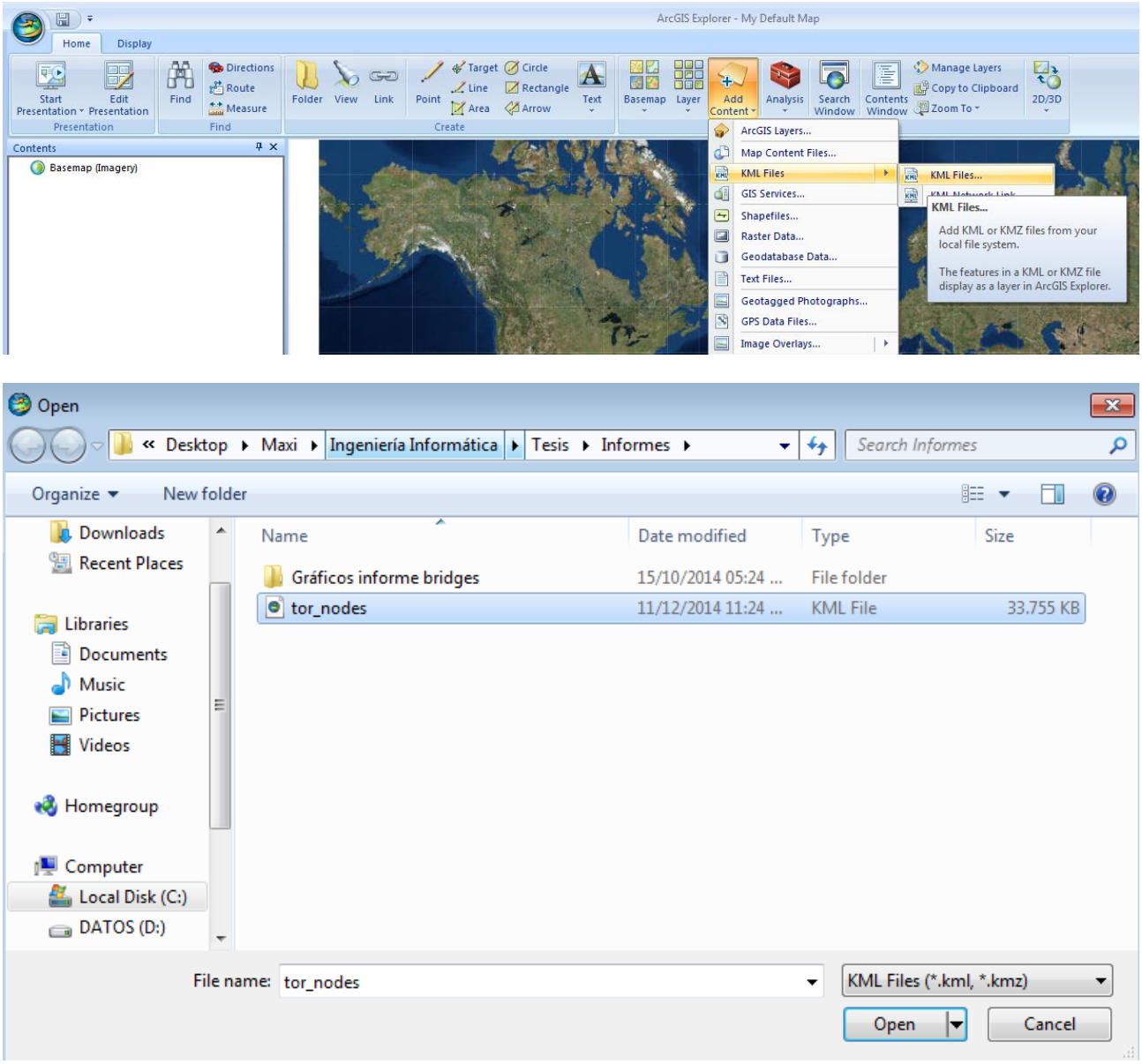


Figura 71: Carga del archivo tor_nodes.kml en ArcGIS

Una vez cargado el archivo KML podemos ver como están distribuidos los nodos Tor alrededor del mundo y su correspondiente nombre de host y si clickeamos sobre uno de ellos podemos ver la descripción que en este caso es la dirección IP del nodo.

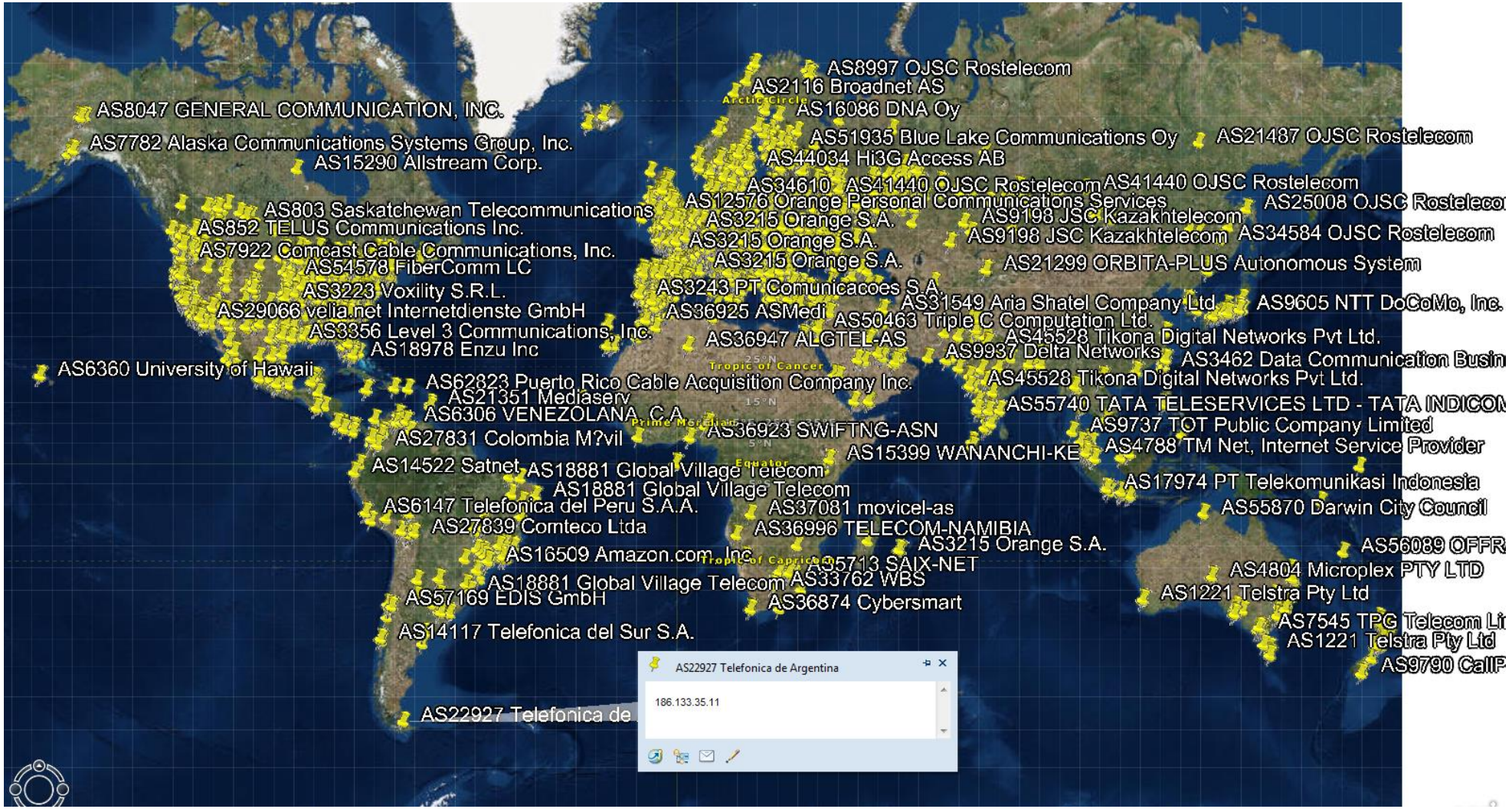


Figura 72: Vista de nodos tor en mapa de ArcGIS

15. MONTAJE DE UN SERVIDOR WEB PARA PRUEBAS DE LABORATORIO

16.1. Introducción

Uno de los puntos vulnerables de nuestra red de la infraestructura crítica del sistema de radar será el web server al que se podrá acceder desde internet y por lo tanto desde cualquier parte del mundo. Por este motivo es que sería conveniente conocer quienes lo acceden y si es una dirección IP de la red TOR, comprobar que no tendrá acceso.

16.2. Instalación Apache Server

Vamos a publicar nuestra página web a través de los servicios que nos brinda Apache Server. Para instalarlo ejecutamos el siguiente comando:

 # yum install httpd

Si tenemos éxito veremos un mensaje como el siguiente:

```
Instalado:
  httpd.i686 0:2.4.6-2.fc18

Dependencia(s) instalada(s):
  apr.i686 0:1.4.6-3.fc18                    apr-util.i686 0:1.4.1-6.fc18
  httpd-tools.i686 0:2.4.6-2.fc18

¡Listo!
```

Figura 73: Instalación de httpd

Luego iniciamos el servicio httpd:

 # systemctl start httpd

Verificamos que funcione correctamente el servicio:

```
[root@192 ~]# systemctl status httpd
httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled)
   Active: active (running) since jue 2015-01-15 20:48:33 ART; 2s ago
 Main PID: 1701 (httpd)
   Status: "Processing requests..."
   CGroup: name=systemd:/system/httpd.service
           ├─1701 /usr/sbin/httpd -DFOREGROUND
           ├─1702 /usr/sbin/httpd -DFOREGROUND
           ├─1703 /usr/sbin/httpd -DFOREGROUND
           ├─1704 /usr/sbin/httpd -DFOREGROUND
           ├─1705 /usr/sbin/httpd -DFOREGROUND
           └─1706 /usr/sbin/httpd -DFOREGROUND

ene 15 20:48:33 192.168.1.108 httpd[1701]: AH00558: httpd: Could not reliabl...e
ene 15 20:48:33 192.168.1.108 systemd[1]: Started The Apache HTTP Server.
```

Figura 74: Estado del servicio httpd

Ahora abrimos un browser en nuestro server y verificamos si está publicada, en localhost, la página por defecto de Apache.

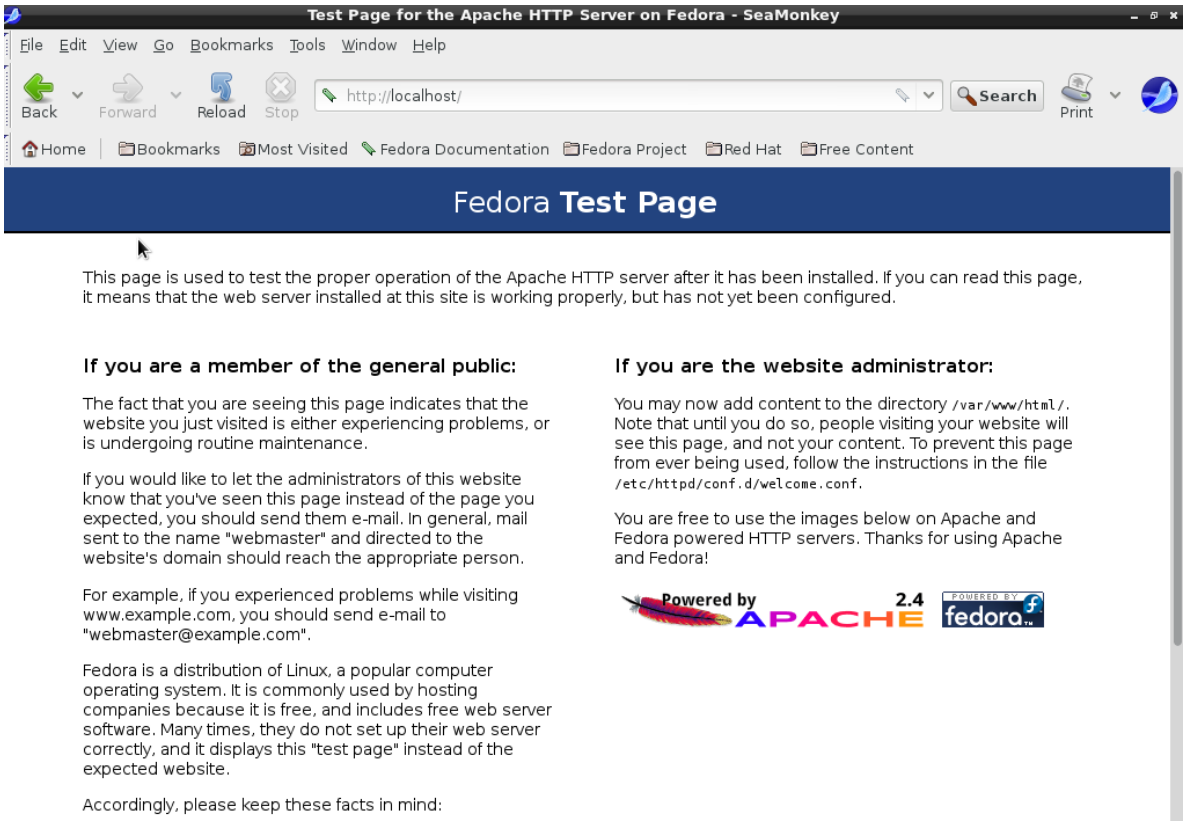


Figura 75: Página por defecto de Apache

16.3. Configuración del servidor virtual

Ahora configuramos un servidor virtual en nuestro router del ISP. Vamos a configurar el Web Server en el puerto 80, con protocolo TCP y la dirección dentro de la LAN es 192.168.1.6:

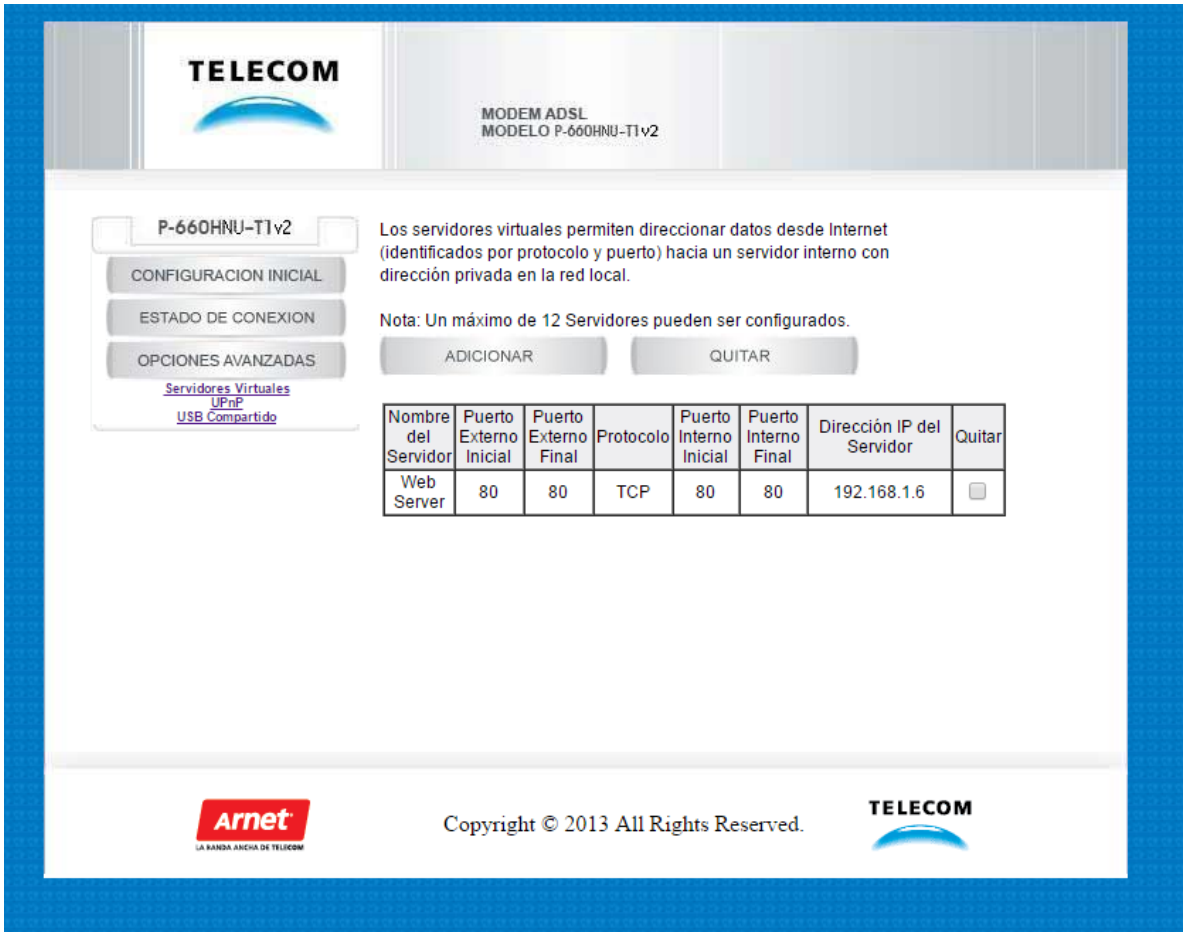


Figura 76: Configuración del servidor virtual en router del ISP

Nuestra dirección pública con la que accederemos desde internet es 181.1.190.175:

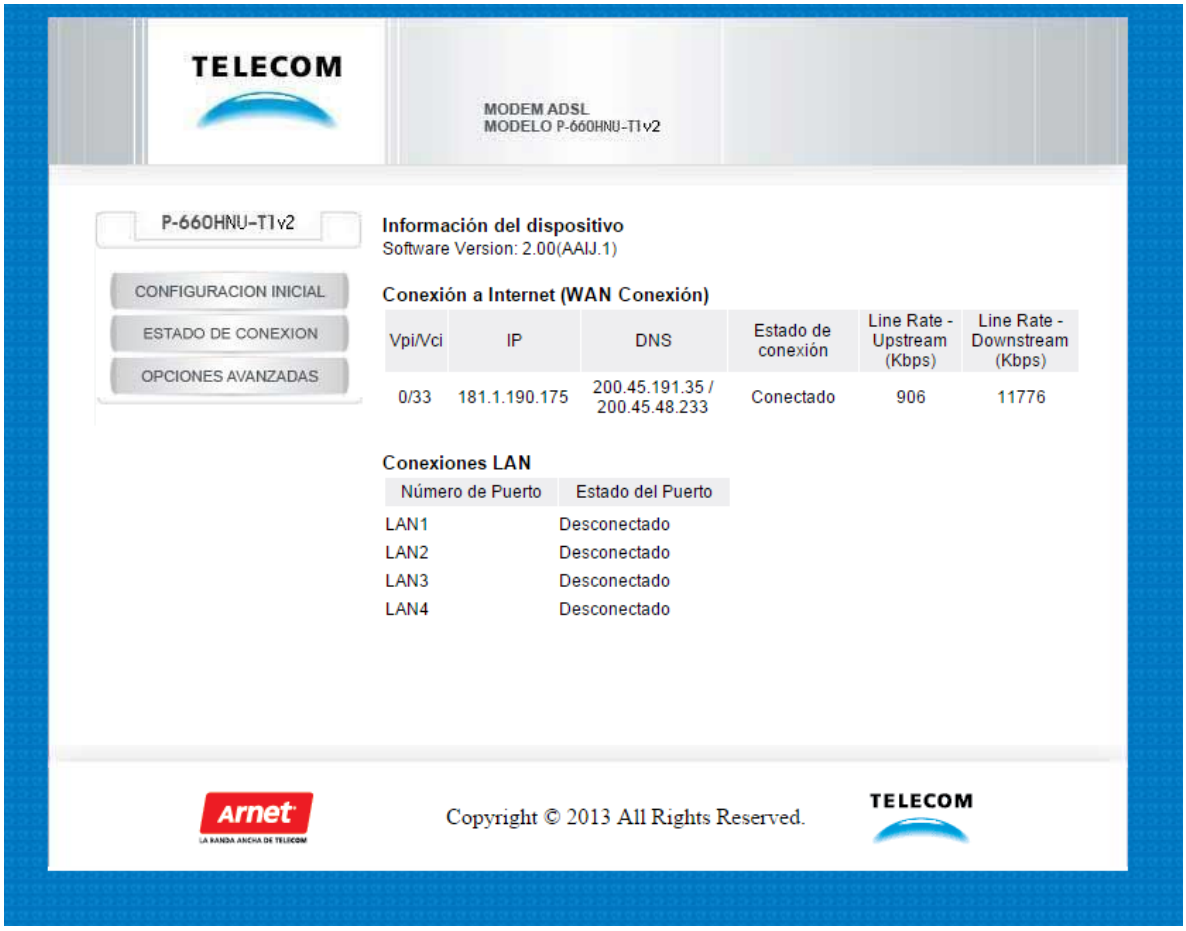


Figura 77: Configuración IP del router del ISP

16.4. Aplicación de seguridad en Web Server

Verificamos que podemos conectarnos exitosamente al servidor a través de la red Tor. Podemos ver que tenemos asignada la IP 188.138.1.229 de esta red:

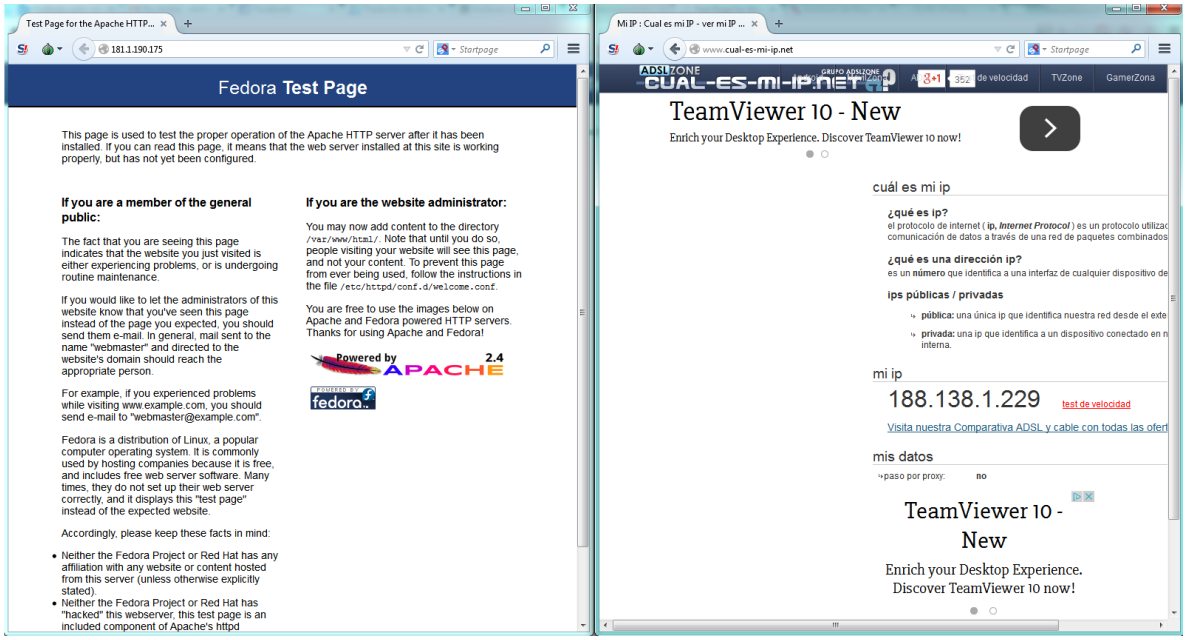


Figura 78: Conexión desde Tor al servidor web exitosa

Para aplicar la seguridad a nuestro servidor debemos modificar el script **iptables_tor_black_list.sh**. La modificación consiste en cambiar la cadena **FORWARD** de las reglas iptables por la cadena **INPUT** ya que se trata de un **Web Server - Firewall** y no un **Router - Firewall** como se trató anteriormente.

De esta manera al ejecutar el script y una vez cargadas las reglas iptables correspondientes, al intentar conectarnos al servidor web desde Tor Browser no lo podemos hacer:

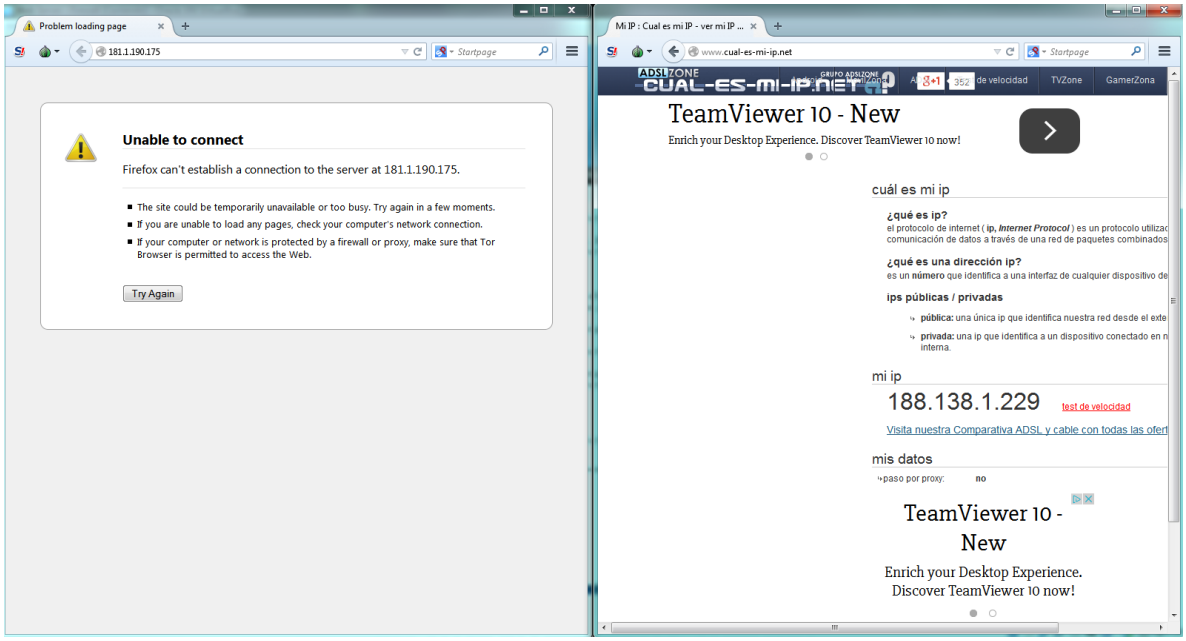


Figura 79: Conexión desde Tor al servidor web fallida

16.5. Ataque DoS sin anonimizar

En el archivo **/etc/httpd/logs/access_log** podemos ver quién se conectar a nuestro servidor. En este caso podemos ver que desde la IP 190.210.81.148 se está intentando realizar un ataque de Denegación de Servicio (DoS):

```
190.210.81.148 - - [17/Jan/2015:13:14:30 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:30 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:30 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:30 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:30 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:30 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:30 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:30 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:30 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:30 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:30 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:30 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:30 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:30 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:30 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:30 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
190.210.81.148 - - [17/Jan/2015:13:14:31 -0300] "GET / HTTP/1.0" 403 4609 "-" "-"
```

Figura 80: Contenido del archivo log del servidor web

Podemos ubicar el origen del ataque con la dirección IP del atacante. Vamos a utilizar el servicio de la página **ipinfo.io** para obtener los siguientes resultados:

190.210.81.148

Hostname	No Hostname
Network	AS16814 S.A.
City	Cordoba, Cordoba, Argentina
Latitude/Longitude	-31.4000,-64.1833
Postal Code	5000

Try the JSON API from the command line

```
$ curl ipinfo.io/190.210.81.148
{
    "ip": "190.210.81.148",
    "hostname": "No Hostname",
    "city": "Cordoba",
    "region": "Cordoba",
    "country": "AR",
    "loc": "-31.4000,-64.1833",
    "org": "AS16814 S.A.",
    "postal": "5000"
}
```

Figura 81: Datos relacionados a la IP pública

Con las coordenadas podemos identificar exactamente el lugar donde se originó el ataque (la dirección donde está declarado el servidor del ISP):

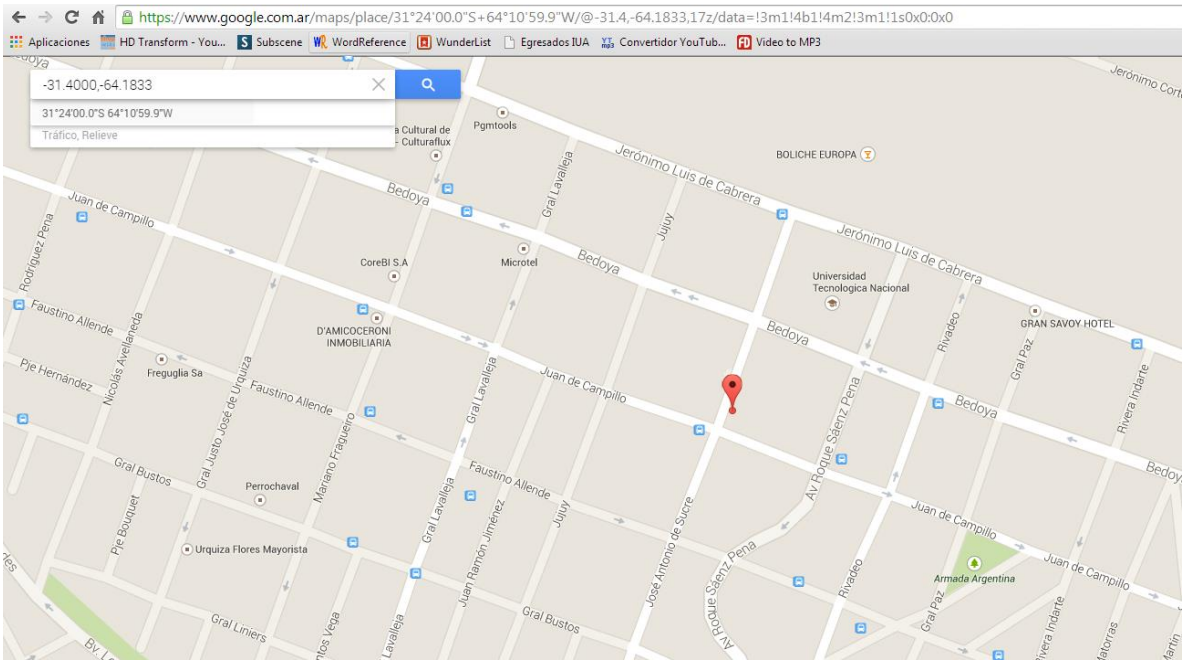


Figura 82: Ubicación física del proveedor de Internet del atacante

16. APLICACIÓN DE LA INVESTIGACIÓN REALIZADA A LA INFRAESTRUCTURA CRÍTICA

17.1. Introducción

En este apartado se mostrará como montar una aplicación web simple de autenticación de usuarios contra una base de datos. La aplicación se va a desplegar sobre un servidor Apache Tomcat y se harán pruebas para ataques desde un browser común y desde Tor Browser.

El diagrama de la infraestructura sobre la que se está trabajando es el siguiente:

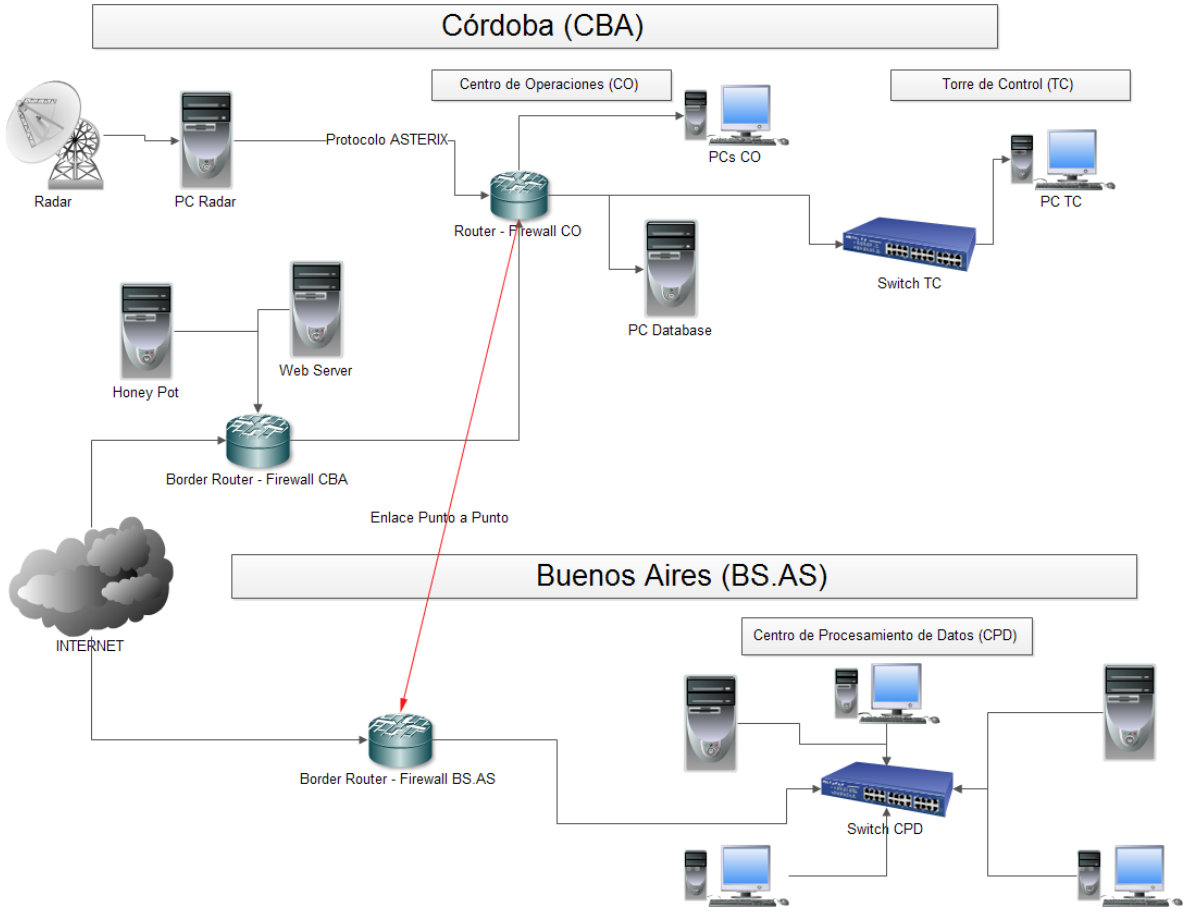


Figura 83: Estructura de red del Sistema de Radarización de Tráfico Aéreo

17.2. Instalación Eclipse Luna

Vamos a trabajar con **Eclipse Luna**, el mismo se puede descargar desde la página https://eclipse.org/downloads/.

Una vez descargado el paquete lo extraemos y Eclipse ya está listo para usarse.

17.3. Instalación de MySQL

Para instalar todos los paquetes relacionados a **MySQL**, ejecutamos el comando:

 # yum install mysql*

Si tenemos éxito veremos el siguiente mensaje:

```
Instalado:
  mysql++.i686 0:3.1.0-10.fc18
  mysql++-devel.i686 0:3.1.0-10.fc18
  mysql++-manuals.i686 0:3.1.0-10.fc18
  mysql-bench.i686 0:5.5.35-1.fc18
  mysql-connector-c++-devel.i686 0:1.1.2-1.fc18
  mysql-connector-java.noarch 1:5.1.22-1.fc18
  mysql-connector-odbc.i686 0:5.1.11-1.fc18
  mysql-connector-python3.noarch 0:1.0.12-1.fc18
  mysql-devel.i686 0:5.5.35-1.fc18
  mysql-embedded.i686 0:5.5.35-1.fc18
  mysql-embedded-devel.i686 0:5.5.35-1.fc18
  mysql-mmm.noarch 0:2.2.1-5.fc18
  mysql-mmm-agent.noarch 0:2.2.1-5.fc18
  mysql-mmm-monitor.noarch 0:2.2.1-5.fc18
  mysql-mmm-tools.noarch 0:2.2.1-5.fc18
  mysql-proxy.i686 0:0.8.1-4.fc18
  mysql-proxy-devel.i686 0:0.8.1-4.fc18
  mysql-server.i686 0:5.5.35-1.fc18
  mysql-test.i686 0:5.5.35-1.fc18
  mysql-workbench.i686 0:5.2.47-2.fc18
  mysqlreport.noarch 0:3.5-7.fc18
  mysqltuner.noarch 0:1.2.0-3.fc18
  mysqludf_xql.i686 0:1.0.0-4.fc18
```

Figura 84: Instalación de MySQL

Luego debemos ejecutar los siguientes comandos para habilitar e iniciar el servicio **mysqld**:

```
[root@192 ~]# systemctl enable mysqld
ln -s '/usr/lib/systemd/system/mysqld.service' '/etc/systemd/system/multi-user.t
arget.wants/mysqld.service'
[root@192 ~]# systemctl start mysqld
```

Figura 85: Habilitación e inicio del servicio mysqld

17.4. Creación de un servidor en Eclipse

Para crear un servidor en Eclipse vamos a **File -> New -> Other…** y seleccionamos **Server -> Server**:

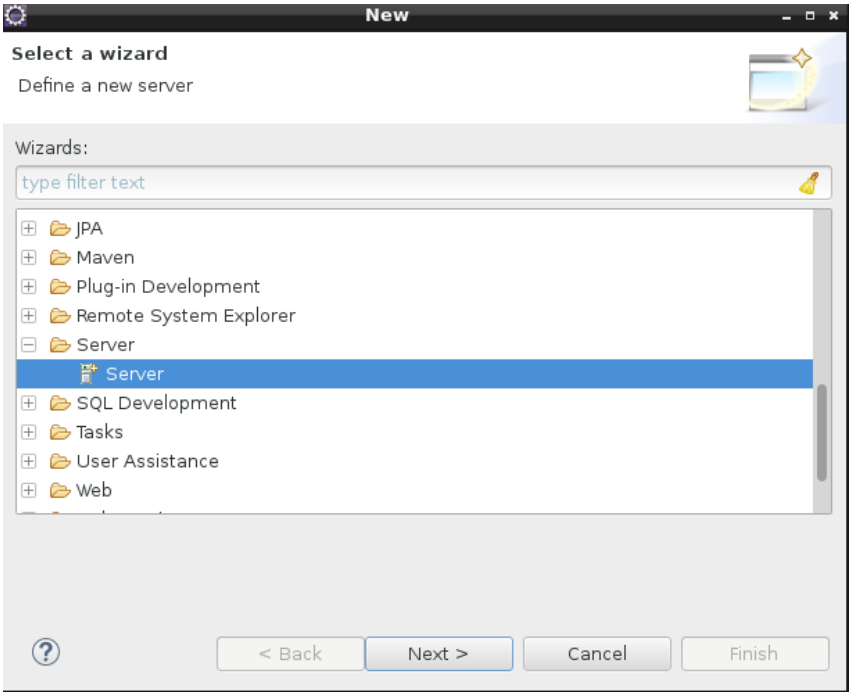


Figura 86: Creación de un servidor dentro de Eclipse 1

Presionamos **Next** y seleccionamos **Apache -> Tomcat v7.0 Server**.

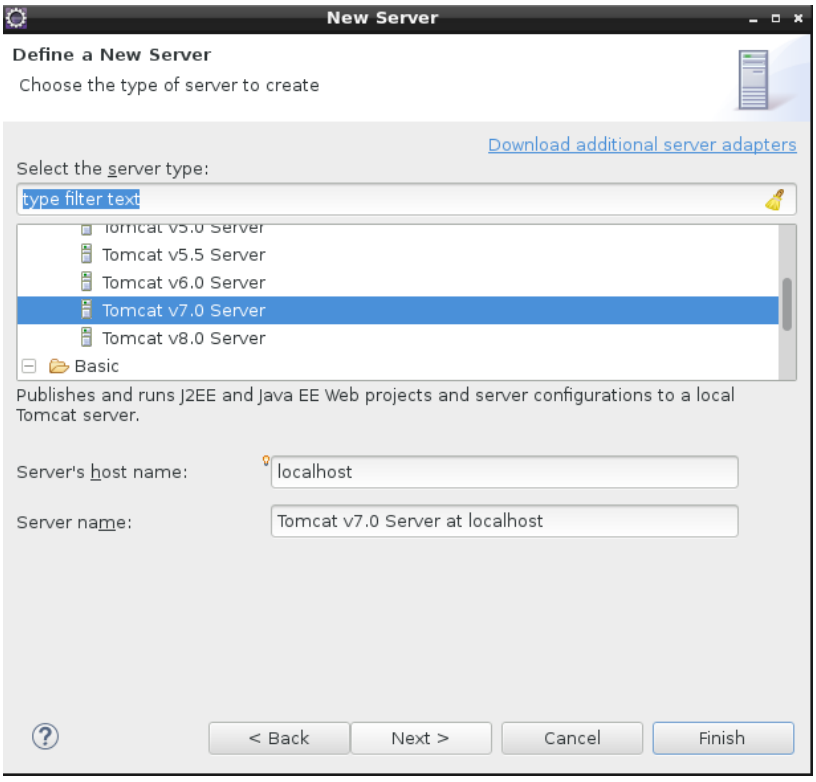


Figura 87: Creación de un servidor dentro de Eclipse 2

Ahora presionamos **Download and Install…**

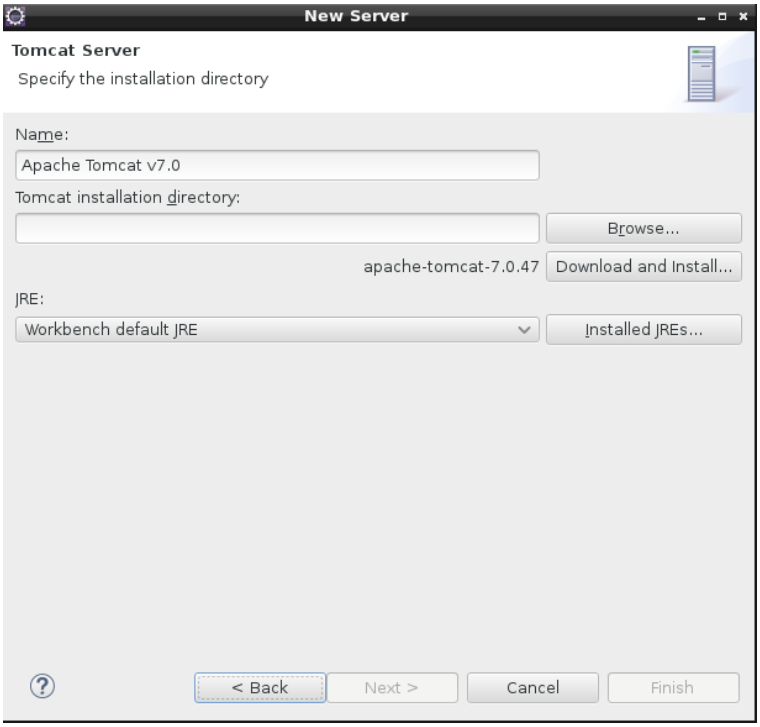


Figura 88: Creación de un servidor dentro de Eclipse 3

Aceptamos los términos de la licencia, presionamos **Finish** y seleccionamos el directorio donde se va a instalar **Apache Tomcat**.

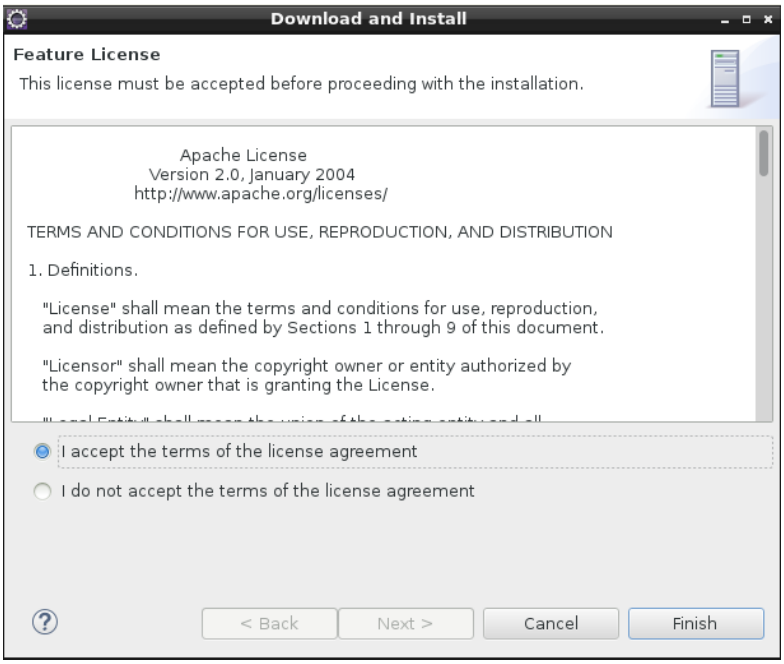


Figura 89: Creación de un servidor dentro de Eclipse 4

Una vez finalizado este proceso, en **JRE** seleccionamos **java openjdk** y presionamos **Finish**.

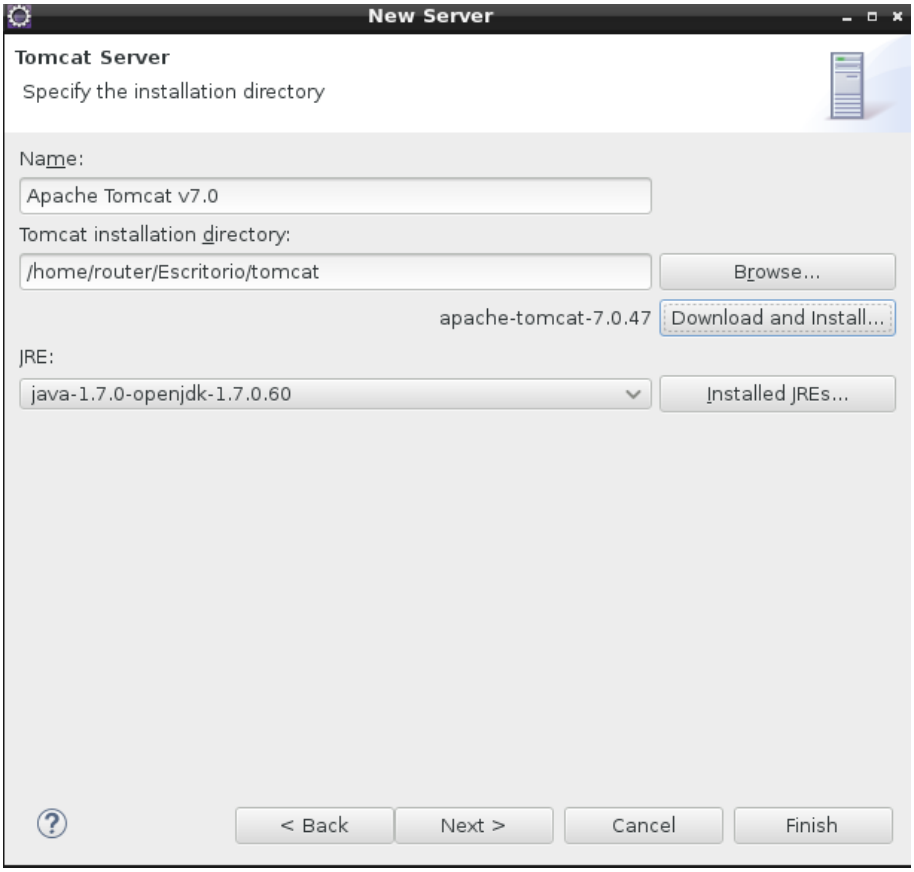
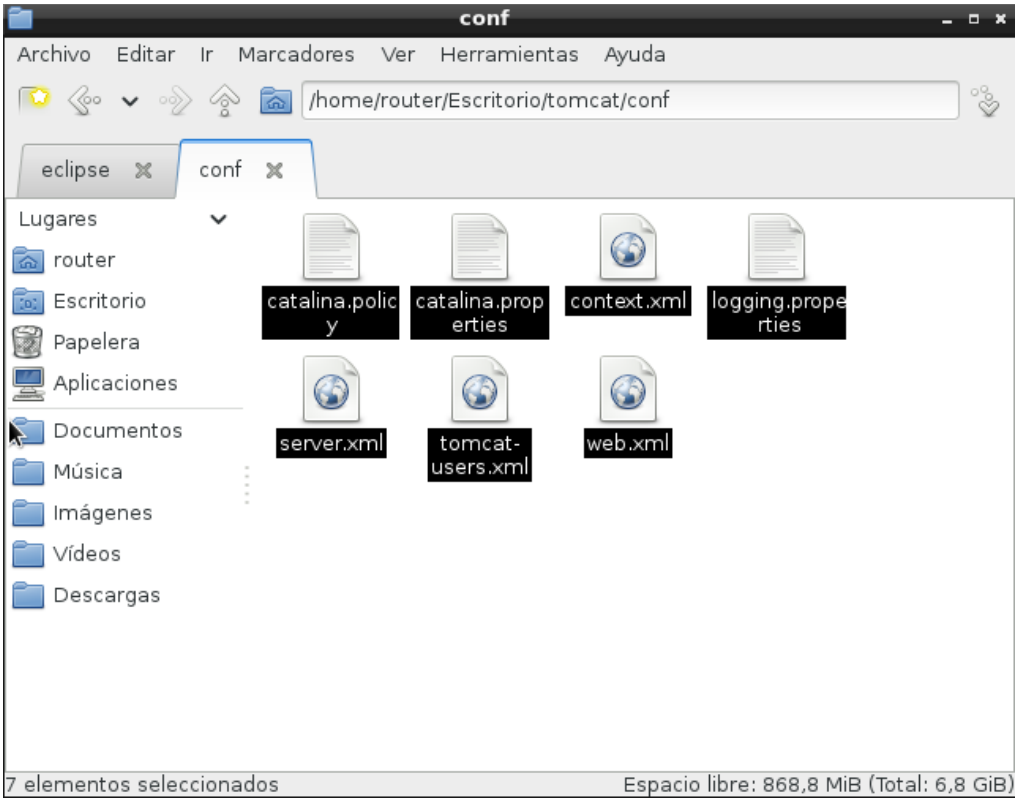


Figura 90: Creación de un servidor dentro de Eclipse 5

Ahora debemos cerrar Eclipse y copiar los archivo de la carpeta **conf** de Tomcat al servidor que está en el **workspace** de Eclipse:



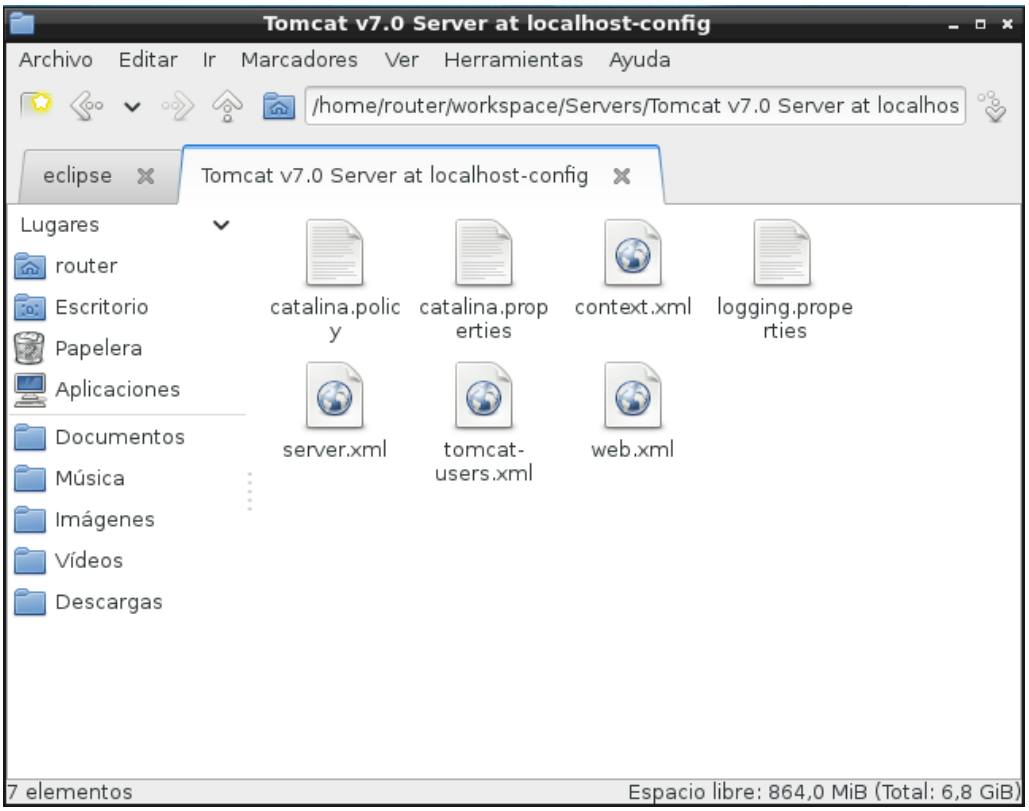


Figura 91: Copia de archivos de configuración de Apache Tomcat

Luego, iniciamos Eclipse nuevamente, vamos al proyecto **Servers** hacemos click derecho en el servidor **Apache Tomcat** y hacemos click en **Refresh**. El próximo paso para iniciar el servidor es ir a la pestaña **Servers**, hacemos click derecho en el servidor y luego en **Start**.

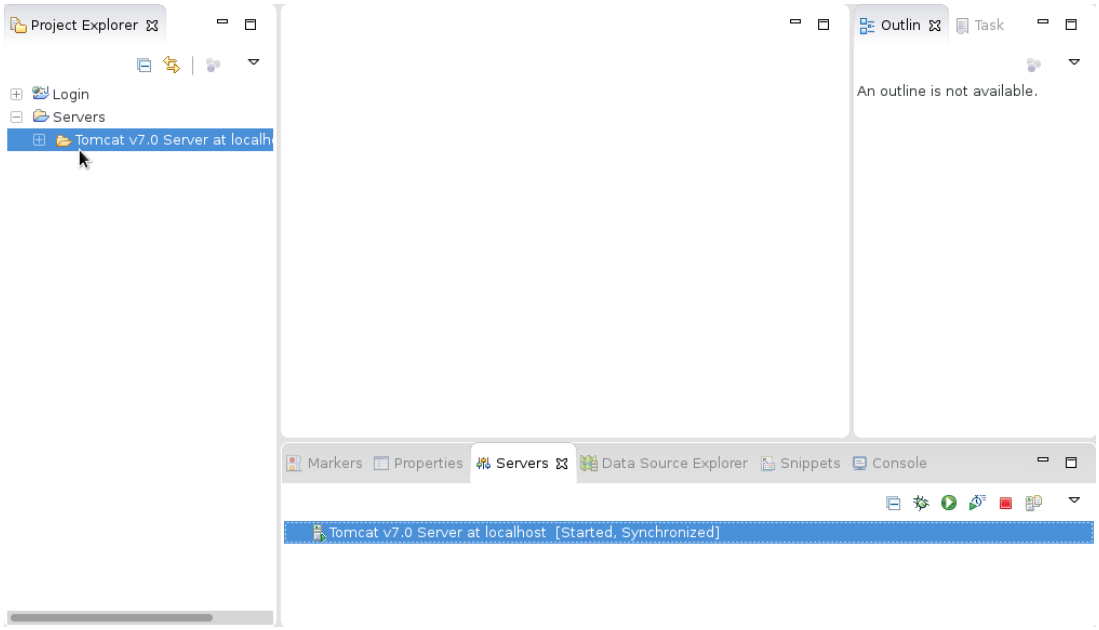


Figura 92: Creación de un servidor dentro de Eclipse 6

17.5. Creación de un proyecto web dinámico en Eclipse

Para crear un proyecto web dinámico en Eclipse vamos a **File -> New -> Other…** y seleccionamos **Web -> Dynamic Web Project**:

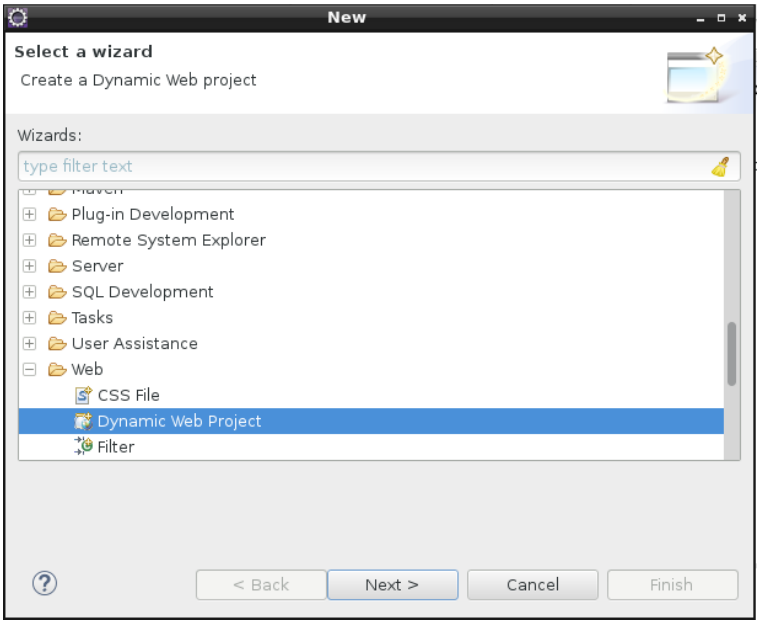


Figura 93: Creación de un proyecto web dinámico en Eclipse 1

Presionamos **Next**, le damos al proyecto el nombre de **Login** y presionamos finalizar.

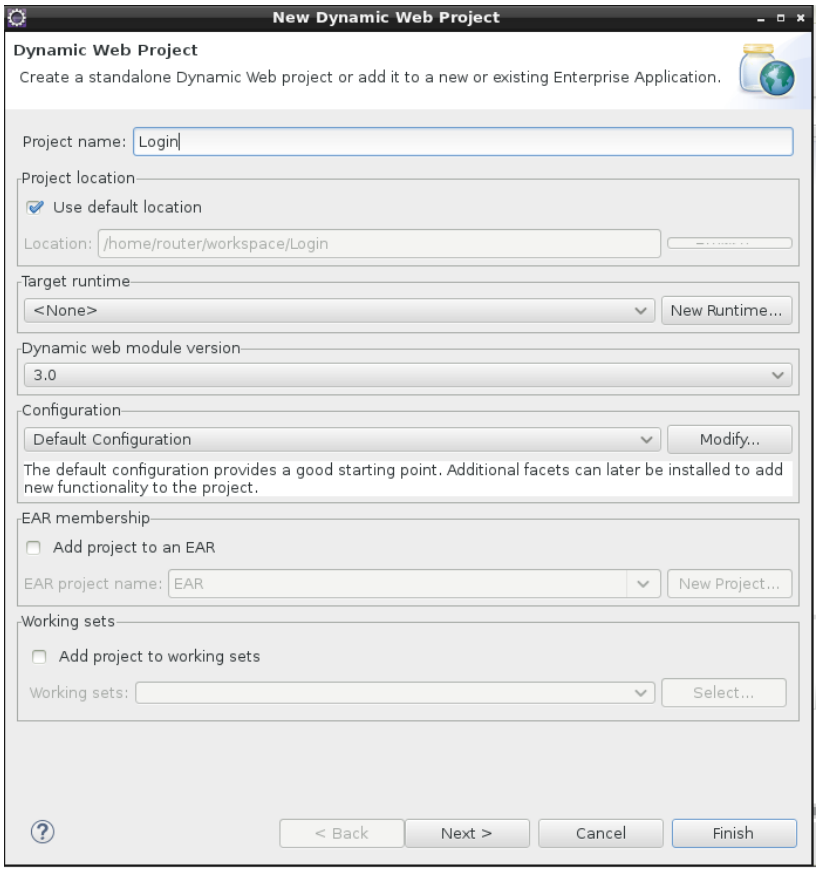


Figura 94: Creación de un proyecto web dinámico en Eclipse 2

Si obtenemos el siguiente error:

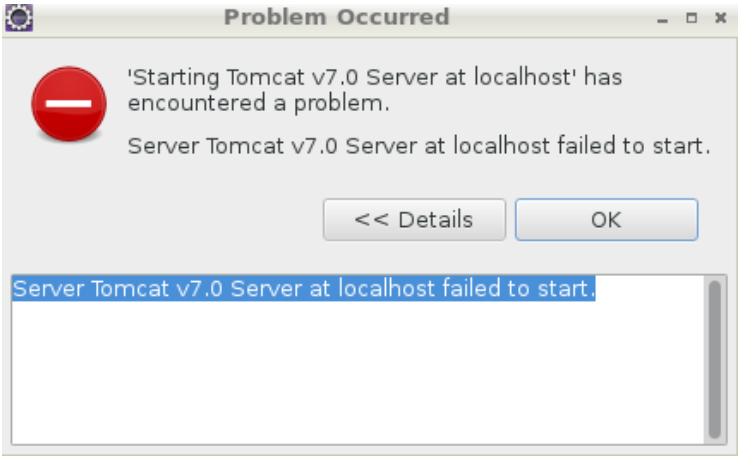


Figura 95: Error al iniciar Apache Tomcat en Eclipse

Lo que debemos hacer es ingresar a la siguiente ruta y eliminar el o los archivos .snap:

```
<workspace-directory>\.metadata\.plugins\org.eclipse.core.resources
```

Figura 96: Directorio contenedor de archivos .snap

Luego reiniciamos Eclipse y ya se puede ejecutar correctamente el proyecto.

Por último vamos a agregar en la carpeta WEB-INF/lib del proyecto el jar de mysql-connector y ya podremos disponer de la conexión desde el proyecto a la base de datos cuando sea necesario.

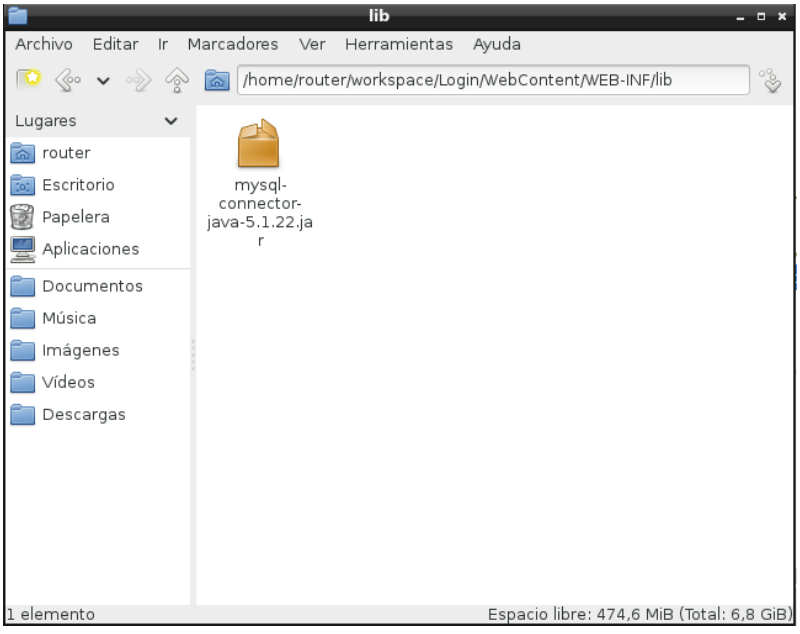


Figura 97: Inclución de mysql-connector-java-5.1.22.jar en el directorio lib de Login

17.6. Creación de la base de datos en MySQL

Abrimos MySQL Workbench y hacemos click en **New Connection**:

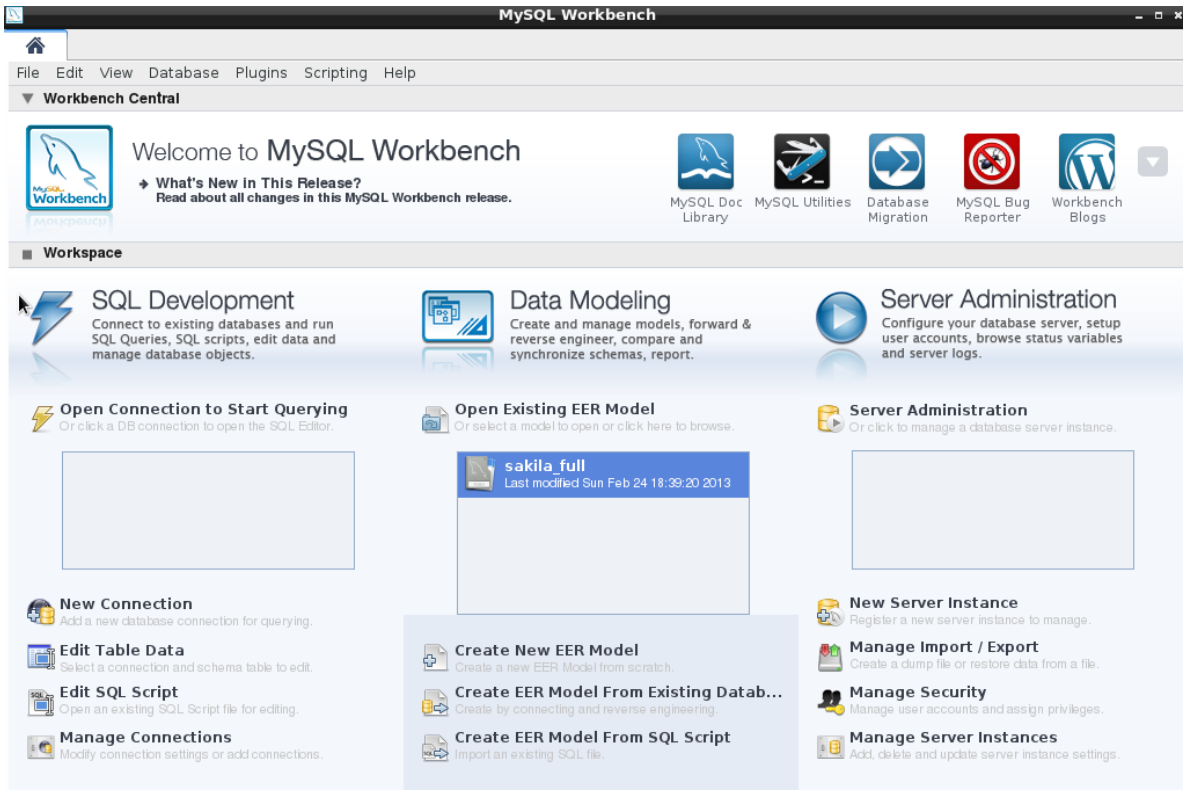


Figura 98: MySQL Workbench

Colocamos el nombre de la conexión, en este caso **login**, presionamos **Test Connection** y si no hay problemas presionamos **OK**:

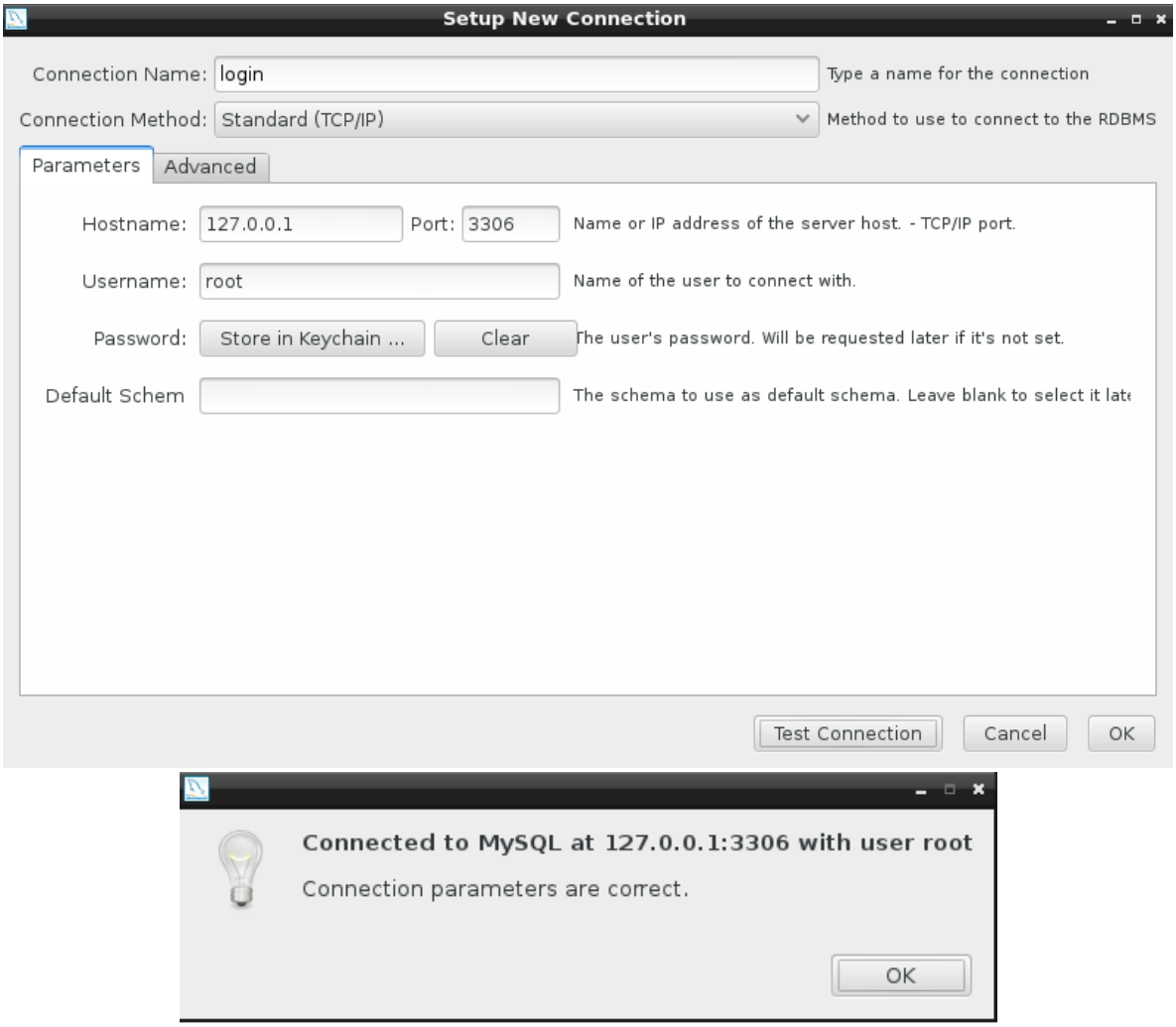


Figura 99: Creación de una conección en MySQL Workbench

Luego abrimos la conexión y crearemos la base de datos **login**. Para ello crearemos un esquema:

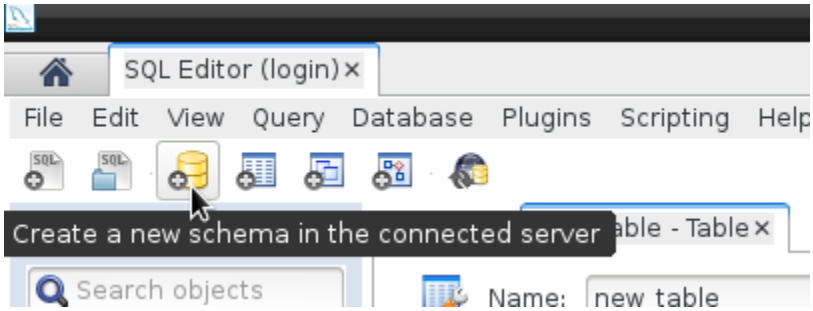


Figura 100: Creación de la base de datos Login 1

Colocamos el nombre de la base de datos y presionamos Apply:

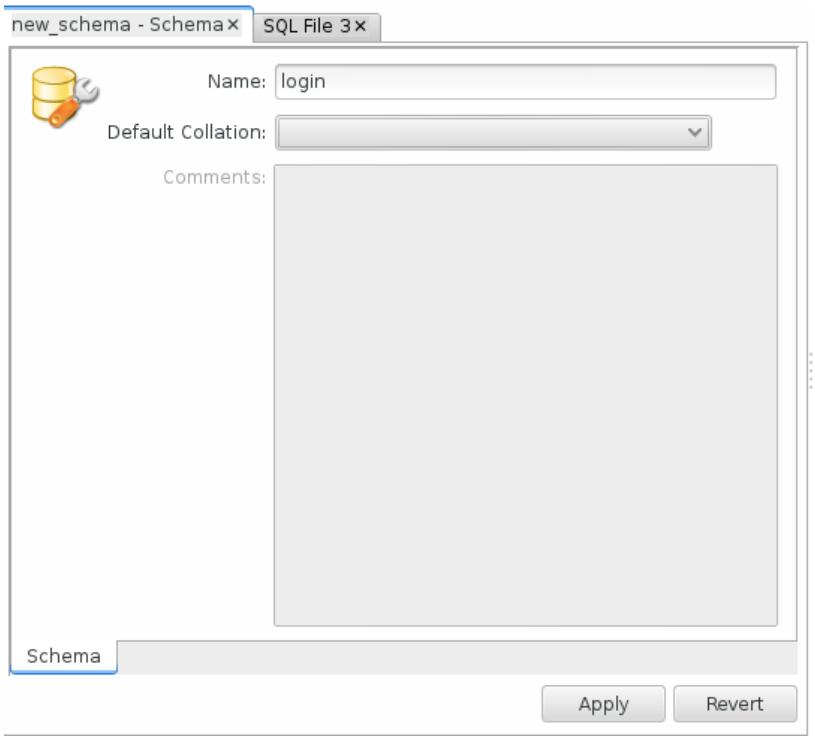


Figura 101: Creación de la base de datos Login 2

Y nuevamente presionamos Apply:

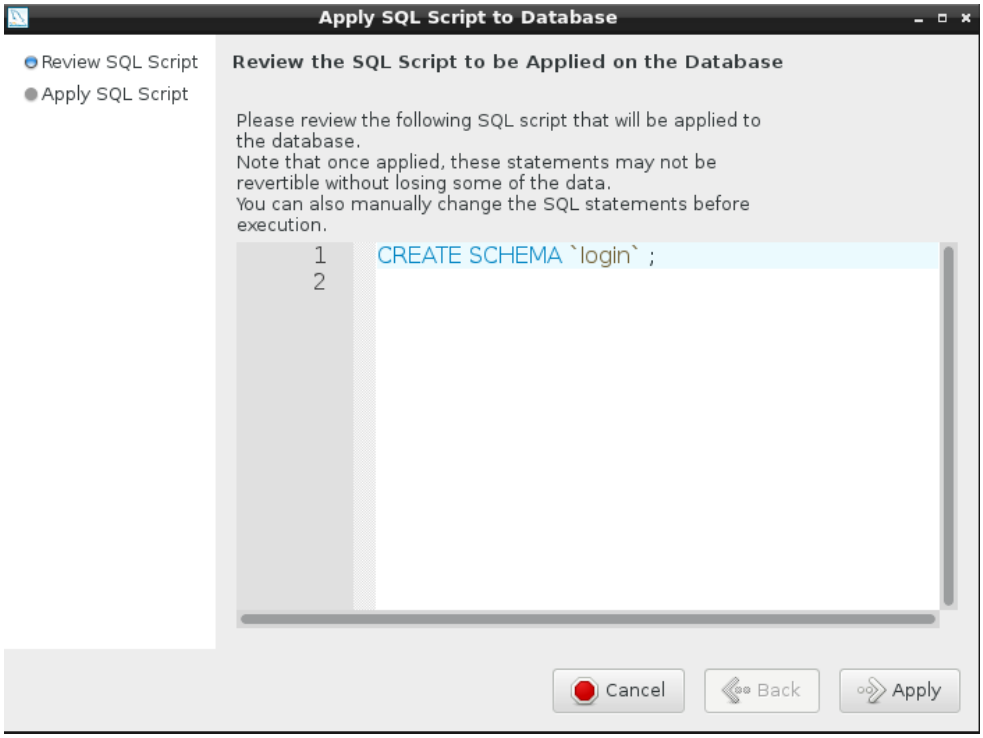


Figura 102: Creación de la base de datos Login 3

Si no hay ningún problema obtendremos el siguiente mensaje y presionamos **Close**:

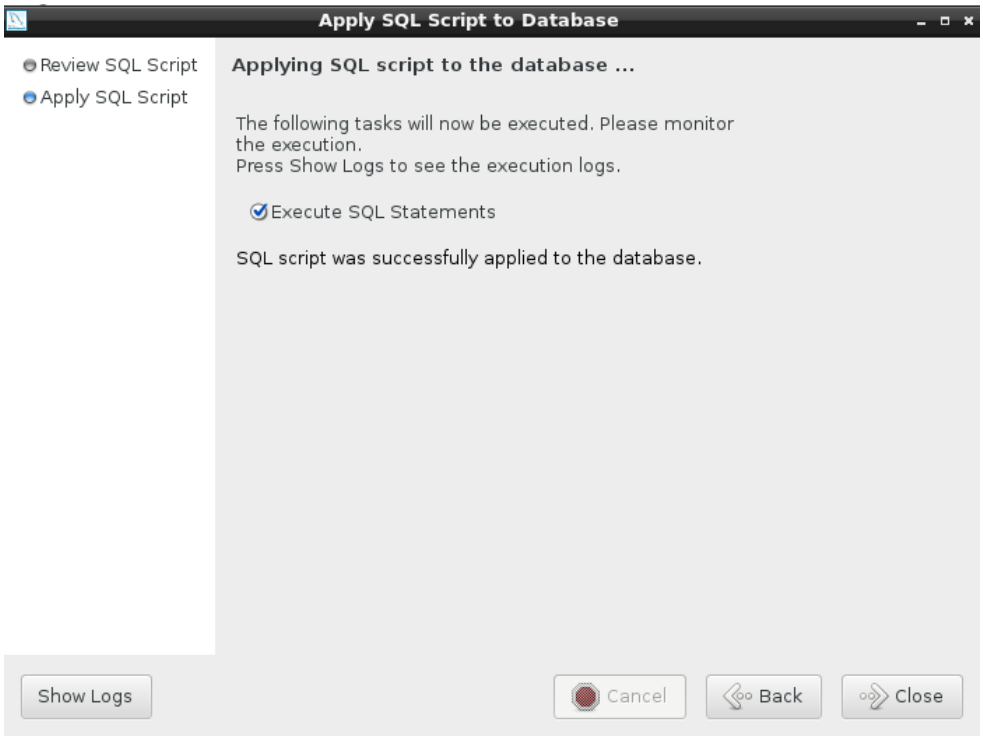


Figura 103: Creación de la base de datos Login 4

Luego crearemos dos tablas en la base de datos, **admin_users** y **users**. Para crear una tabla desplegamos la base de datos **login**, hacemos click en **Tables -> Create Table…**. Cada tabla presentará la siguiente configuración:

Tabla **admin_users**:

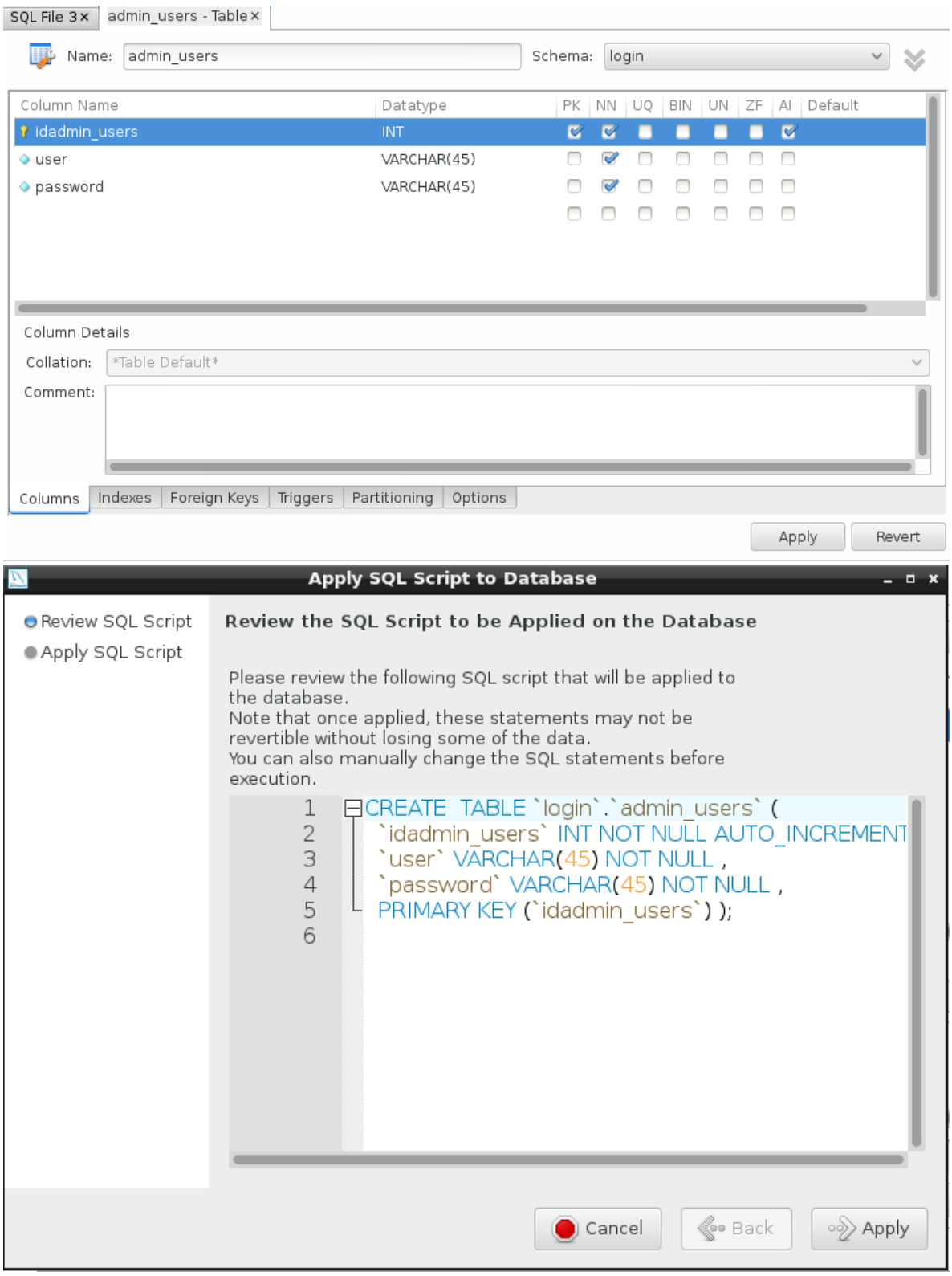


Figura 104: Creación de la tabla admin_users

Tabla **users**:

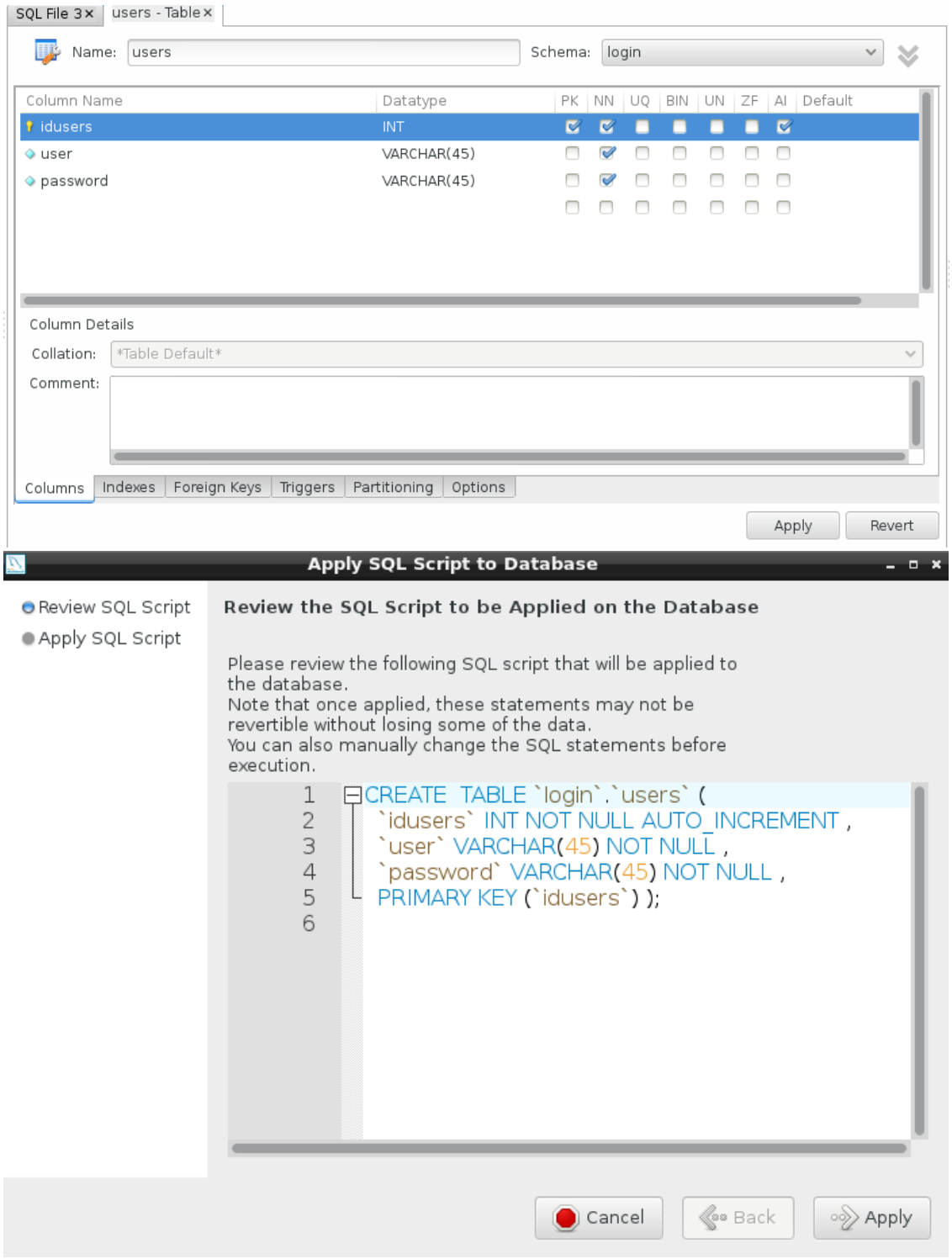


Figura 105: Creación de la tabla users

Contenido de la tabla **admin_users**:

#	idadmin_users	user	password
1	1	master	master951!.
2	2	admin	Jorge441h1

Figura 106: Contenido de la tabla admin_users

Contenido de la tabla **users**:

#	idusers	user	password
1	1	operador	op1995zk1
2	2	usuario1	15936482

Figura 107: Contenido de la tabla users

17.7. Contenido del proyecto web Login

El proyecto web Login, consiste simplemente en una página jsp de inicio de sesión, la cual valida a través de un JavaBean los datos ingresados y luego redirige al usuario a una página de acceso o de falla. A continuación vemos el código más importante del proyecto.

Página **index.jsp**:

```
1  <%@page contentType="text/html" pageEncoding="UTF-8"%>
2  <%@page import = "sesion.BeanSesion" %>
3  <jsp:setProperty name="sesion" property="*" />
4  <jsp:useBean id="sesion" class="sesion.BeanSesion" scope="session" >
5      </jsp:useBean>
6  <!DOCTYPE html>
7  <html>
8  <head>
9  <title>Iniciar Sesión</title>
267 </head>
268
269 <body>
270     <form id="login" action="index.jsp">
271     <h1>INGRESO</h1>
272        <fieldset id="inputs">
273          <input id="username" type="text" name="user" placeholder="Usuario" autofocus required>
274          <input id="password" type="password" name="pass" placeholder="Contraseña" required>
275     </fieldset>
276     <fieldset id="actions">
277         <input type="submit" id="submit" value="Iniciar Sesión">
278     </fieldset>
279 </form>
280 </body>
281 </html>
282 <% if(sesion.iniciarSesion().equals("entrar")){ %>
283     <jsp:forward page="acceso.jsp"/>
284 <% }else if(sesion.iniciarSesion().equals("falla")){%>
285     <jsp:forward page="falla.jsp"/>
286 <% } %>
```

Figura 108: Código de la página index.jsp

Página **acceso.jsp**:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Acceso</title>
</head>
<body>
    <h1>Accediste!</h1>
</body>
</html>
```

Figura 109: Código de la página acceso.jsp

Página **falla.jsp**:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Falla</title>
</head>
<body>
    <h1>El usuario y contraseña no son correctos</h1>
</body>
</html>
```

Figura 110: Código de la página falla.jsp

JavaBean **BeanSesion**:

El método **iniciarSesion** se encarga de realizar la validación de los datos ingresados por el usuario contra la base de datos.

```
    public String iniciarSesion() throws SQLException{
        if(user != null && pass != null){
        try
            {
                Class.forName("com.mysql.jdbc.Driver");
                 } catch (Exception e){
                e.printStackTrace();
            }
            // Establecemos la conexión con la base de datos.
            conn = DriverManager.getConnection("jdbc:mysql://localhost/login","root","");
            // Preparamos la consulta
            s = conn.createStatement();
            rs = s.executeQuery ("select user, password from users");
            // Recorremos el resultado, mientras haya registros para leer, y escribimos el resultado en par
            while (rs.next()){
                if(rs.getString("user").equals(user) && rs.getString("password").equals(pass)){
                    conn.close();
                    rs.close();
                    s.close();
                    return "entrar";
                }
            }
            rs = s.executeQuery ("select user, password from admin_users");
            while (rs.next()){
                if(rs.getString("user").equals(user) && rs.getString("password").equals(pass)){
                    conn.close();
                    rs.close();
                    s.close();
                    return "entrar";
                }
            }
```

```
66              // Cerramos la conexion a la base de datos.
67              conn.close();
68              rs.close();
69              s.close();
70              return "falla";
71          }
72          return "";
73      }
74  }
```

Figura 111: Código del método iniciarSesion

17.8. Publicación del proyecto Login

El proyecto se desplegará en el servidor Apache Tomcat haciendo click derecho sobre el proyecto **Run As -> Run On Server**. Una vez que se despliegue el proyecto y esté iniciado el servidor, podremos ver la página de inicio del proyecto.

Figura 112: Página de inicio del proyecto Login

Se realizó la misma configuración que se explicó anteriormente para publicar el servidor web local como servidor virtual en el router del ISP y así poder acceder desde la nube al proyecto Login a través de la dirección IP pública que nos brinda nuestro ISP.

17.9. Router – Firewall y Servidor Web

En este caso, y para coincidir con la estructura de la infraestructura crítica planteada, vamos a tener un router-firewall a través del cual tendremos salida a la nube y cargaremos en el las reglas del firewall. Detrás de este router vamos a ubicar a nuestro servidor web.

Se utilizará la misma configuración del router de que se estuvo trabajando en los informes anteriores. El mismo, ejecutará el script **iptables_tor_black_list.sh**, para aplicar la seguridad contra la red Tor cuando sea necesario, y el script **iptables_squid.sh**, para configurar un proxy transparente y para publicar el servidor web en una de las interfaces del router.

La configuración IP del servidor web es la siguiente:

```
HWADDR=08:00:27:1D:FB:47
TYPE=Ethernet
#BOOTPROTO=dhcp
BOOTPROTO=static
IPADDR=192.168.2.45
NETMASK=255.255.255.0
#GATEWAY=192.168.1.1
DEFROUTE=yes
#PEERDNS=yes
PEERDNS=no
PEERROUTES=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
NAME=p2p1
UUID=3b6dc792-690d-481d-b788-fe4ff0d7da2b
ONBOOT=yes
```

Figura 113: Configuración IP del servidor web

Configuración IP de la interfaz **p2p1** del router:

```
HWADDR=08:00:27:F8:4B:49
TYPE=Ethernet
#BOOTPROTO=dhcp
BOOTPROTO=static
IPADDR=192.168.1.150
NETMASK=255.255.255.0
GATEWAY=192.168.1.1
DEFROUTE=yes
#PEERDNS=yes
PEERDNS=no
PEERROUTES=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
NAME=p2p1
UUID=3b6dc792-690d-481d-b788-fe4ff0d7da2b
ONBOOT=yes
```

Figura 114: Configuración IP de la interfaz p2p1 del router

Configuración IP de la interfaz **p7p1** del router:

```
HWADDR=08:00:27:51:B5:7F
TYPE=Ethernet
#BOOTPROTO=dhcp
BOOTPROTO=static
IPADDR=192.168.2.150
NETMASK=255.255.255.0
DEFROUTE=yes
PEERDNS=no
PEERROUTES=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
NAME=p7p1
UUID=a534a4f1-a19a-4df0-abab-e940937a7621
ONBOOT=yes
```

Figura 115: Configuración IP de la interfaz p7p1 del router

Se realizaron algunas modificaciones en el script **iptables_squid.sh** para poder publicar la página alojada en el servidor web. Podemos ver las reglas del script a continuación:

```
#!/bin/bash
#

iptables -F
iptables -X
iptables -F -t nat
iptables -X -t nat

# Configuracion para publicar el servidor web
iptables -A FORWARD -s 192.168.1.0/24 -i p2p1 -p tcp --dport 80 -j ACCEPT
iptables -t nat -A PREROUTING -i p2p1 -p tcp --dport 80 -j DNAT --to-destination 192.168.2.45:80
iptables -t nat -A POSTROUTING -s 192.168.2.0/24 -o p2p1 -j SNAT --to 192.168.1.150

# Proxy Transparente
iptables -t nat -A POSTROUTING -s 192.168.2.0/24 -o p2p1 -j SNAT --to 192.168.1.150
iptables -t nat -A PREROUTING -i p7p1 -p tcp --dport 80 -j DNAT --to 192.168.1.150:3128
iptables -t nat -A PREROUTING -i p2p1 -p tcp --dport 80 -j REDIRECT --to-port 3128
```

Figura 116: Modificación del script iptables_squid.sh para publicar la web

El script **iptables_tor_black_list.sh** es el mismo que se describió anteriormente. Se exponen sus líneas a continuación:

```
#!/bin/bash
#

total=0
nuevos=0

# Creacion de el directorio contenedor Tor (si no existe)
mkdir -p Tor

# Descarga de la lista actual de nodos y almacenamiento en el archivo Tor_ip_list_ALL
wget http://torstatus.blutmagie.de/ip_list_all.php/Tor_ip_list_ALL.csv -O Tor/Tor_ip_list_ALL

# Ordenamiento de la lista del archivo Tor_ip_list_ALL
sort -t . -k 1,1n -k 2,2n -k 3,3n -k 4,4n Tor/Tor_ip_list_ALL -o Tor/Tor_ip_list_ALL

# Creacion de archivo Tor_black_list (si no existe)
touch Tor/Tor_black_list

# Agregamos los bridges privados a la lista
cat /home/router/Tor_bridges/blacklist_bridges >> Tor/Tor_black_list

# Ordenamiento de la lista del archivo Tor_black_list
sort -t . -k 1,1n -k 2,2n -k 3,3n -k 4,4n Tor/Tor_black_list -o Tor/Tor_black_list

# Posibles direcciones repetidas eliminadas
cp Tor/Tor_ip_list_ALL Tor/aux
cat Tor/aux | uniq > Tor/Tor_ip_list_ALL
rm -f Tor/aux

# Almacenamiento de las nuevas IP en el archivo Tor_black_list
# y carga de nuevas reglas iptables
for nodo in $(diff Tor/Tor_black_list Tor/Tor_ip_list_ALL | grep -E '> ([0-9]{1,3}\.){3}[0-9]{1,3}' | sed 's/^> *//g')
do
        echo $nodo >> Tor/Tor_black_list
        iptables -A FORWARD -s $nodo -j DROP
        ((nuevos++))
done
```

```
# Nuevo ordenamiento de la lista del archivo Tor_black_list
sort -t . -k 1,1n -k 2,2n -k 3,3n -k 4,4n Tor/Tor_black_list -o Tor/Tor_black_list

# Posibles direcciones repetidas eliminadas
cp Tor/Tor_black_list Tor/aux
cat Tor/aux | uniq > Tor/Tor_black_list
rm -r Tor/aux

# Revision de la black list
for nodo in $(cat Tor/Tor_black_list)
do
        iptables -C FORWARD -s $nodo -j DROP 2>/dev/null
        if [ $? -ne 0 ]; then
                iptables -A FORWARD -s $nodo -j DROP
                ((nuevos++))
        fi
        ((total++))
done

echo Nuevos nodos bloqueados: $nuevos
echo Total nodos bloqueados: $total

# Copia para analisis de metricas de Tor_ip_list_ALL
mkdir -p Tor/analisis
cp Tor/Tor_ip_list_ALL Tor/analisis/Tor_ip_list_ALL_$(date +%y%m%d%H%M)

# copia de seguridad de Tor_black_list
mkdir -p Tor/backup_tor
cp Tor/Tor_black_list Tor/backup_tor/Tor_black_list_$(date +%y%m%d%H%M)
```

Figura 117: Script iptables_tor_black_list.sh

Configuramos el servidor web para publicar en el puerto 80. Debemos ingresar en **server.xml** y modificar la siguiente línea:

```
64     <Connector connectionTimeout="20000" port="80" protocol="HTTP/1.1" redirectPort="8443" />
```

Figura 118: Configuración del servidor para publicar en el puerto 80

Creamos un servidor virtual en el router del ISP con la dirección IP de la interfaz p2p1 del router, 192.168.1.150:

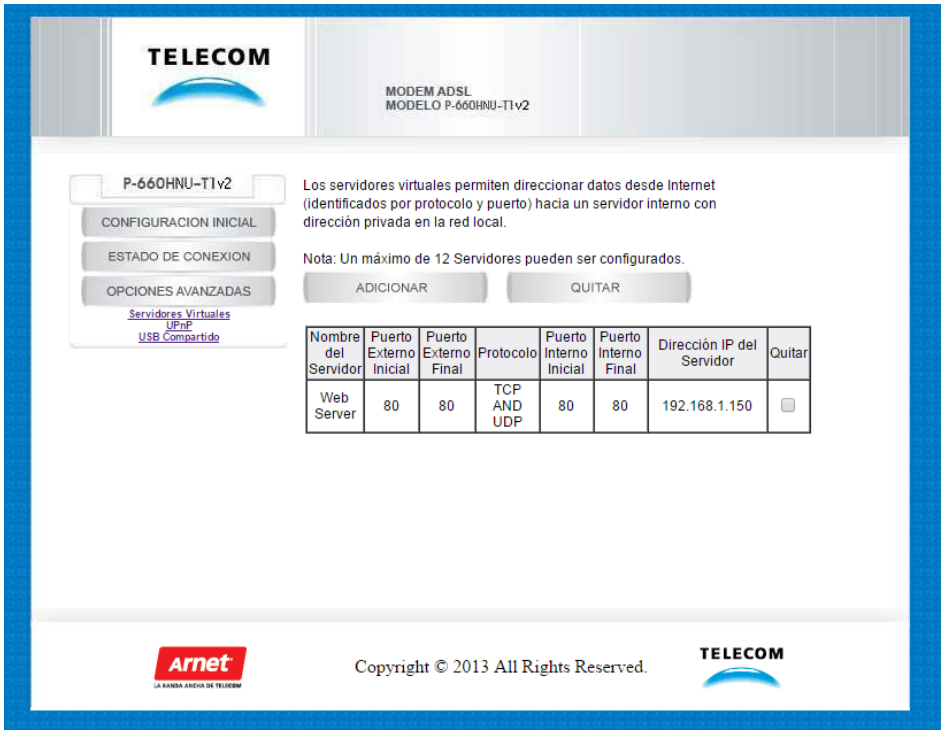


Figura 119: Configuración del servidor virtual en el router del ISP

A continuación podemos ver la dirección IP pública para acceder a la aplicación web:

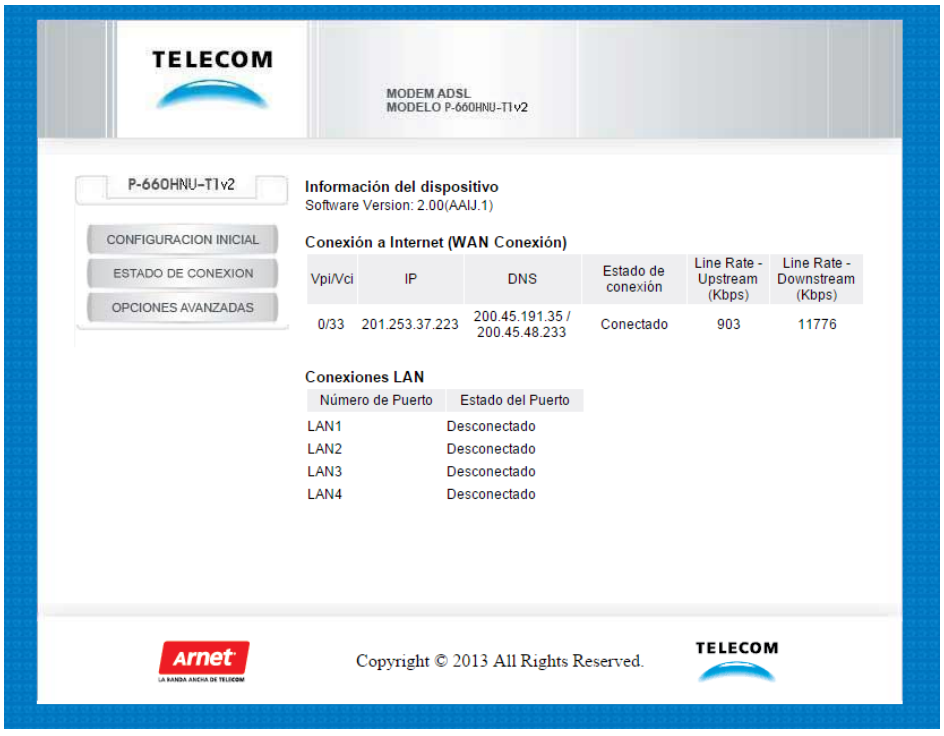


Figura 120: Configuración IP del router del ISP

17.10. Ataque SQLi

Se modificará la forma de obtener los datos de la base de datos en el método iniciarSesion de la clase BeanSesion para hacerlo vulnerable a ataques SQLi y demostrar la gravedad de malas prácticas de programación. El código modificado es el siguiente, en donde se utilizan variables para obtener los datos:

```
rs = s.executeQuery ("select user, password from users where user='"+user+"' and password='"+pass+"'");
// Recorremos el resultado, mientras haya registros para leer, y escribimos el resultado en pantalla.
if (rs.next() == true){
    conn.close();
    rs.close();
    s.close();
    return "entrar";
}
rs = s.executeQuery ("select user, password from admin_users where user='"+user+"' and password='"+pass+
// Recorremos el resultado, mientras haya registros para leer, y escribimos el resultado en pantalla.
if (rs.next() == true){
    conn.close();
    rs.close();
    s.close();
    return "entrar";
}
// Cerramos la conexion a la base de datos.
conn.close();
rs.close();
s.close();
return "falla";
```

Figura 121: Código vulnerable a ataques SQLi

Dada esta vulnerabilidad podemos observar que podemos ingresar de cualquier manera o con cualquier usuario al sistema.

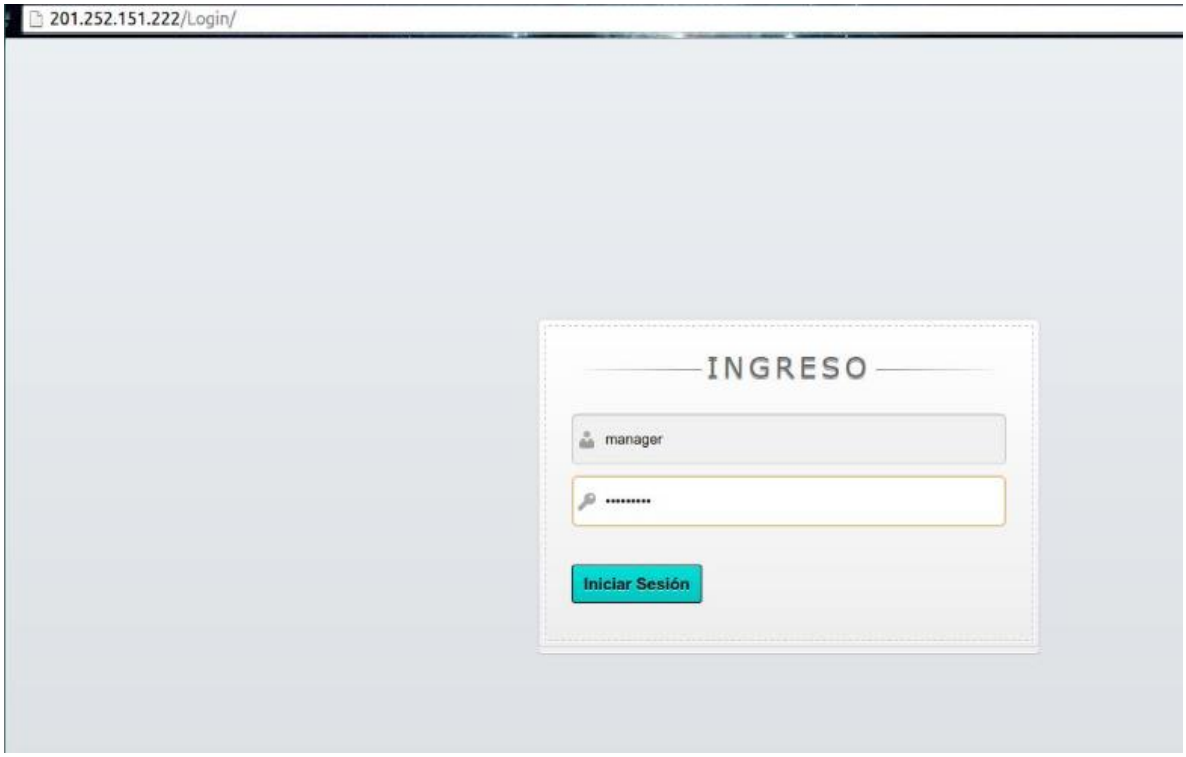


Figura 122: Ataque SQLi al servidor web desde un browser común 1

Intentaremos acceder con el usuario **master** pero en la contraseña colocaremos **' OR ''='**. De esta forma, cuando se ejecute la consulta, será siempre verdadera y nos dará acceso al sistema.

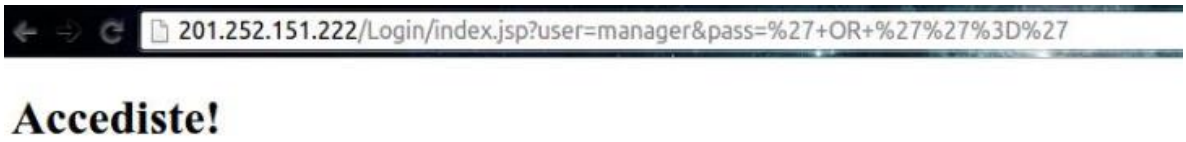


Figura 123: Ataque SQLi al servidor web desde un browser común 2

17.11. Ataque SQLi desde TOR

Vamos a realizar el mismo ataque desde Tor Browser y comprobaremos que podemos acceder:

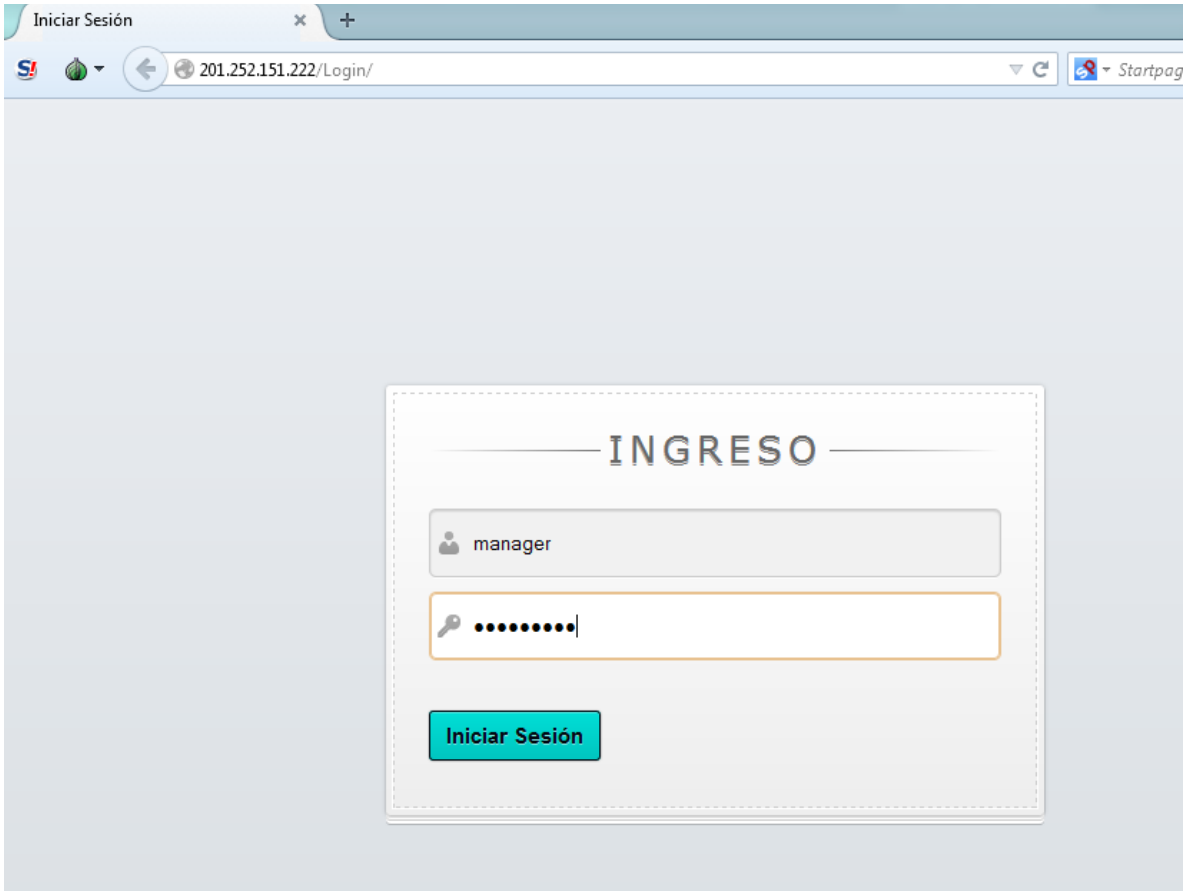


Figura 124: Ataque SQLi al servidor web desde Tor Browser 1

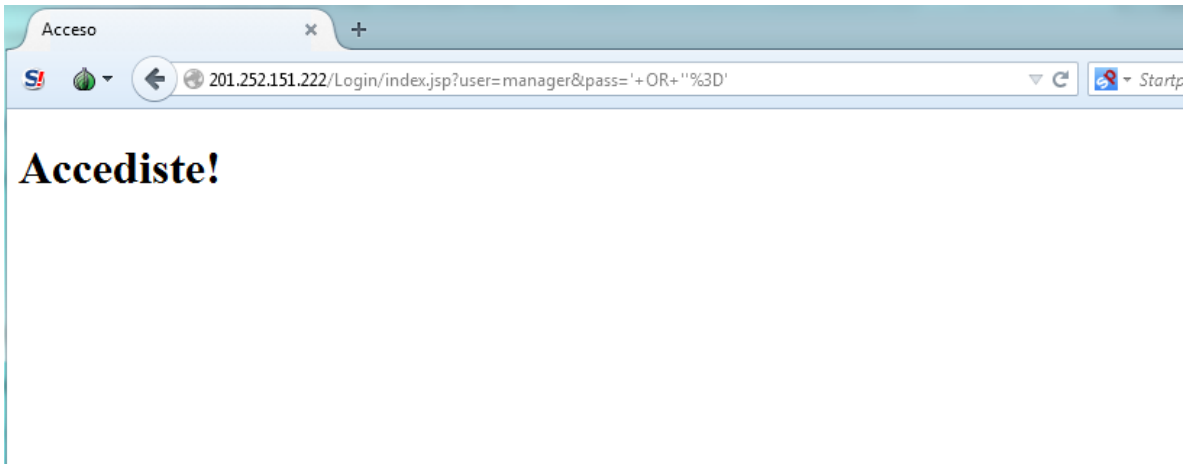


Figura 125: Ataque SQLi al servidor web desde Tor Browser 2

La ventaja que tenemos aquí es que no se conocerá nuestra verdadera identidad.

17.12. Detectar el origen de los ataques

De todas formas, al concretarse el ataque y una vez detectado, podemos revisar los logs de acceso de Apache Tomcat y obtener la dirección IP desde donde se realizó el ataque. Los logs de Tomcat, si estamos utilizando el servidor de Eclipse, se encuentran por defecto en **$workspace/.metadata/.plugins/org.eclipse.wst.server.core/temp0/logs**.

```
-rw-rw-r-- 1 router router 3223 ene 27 22:05 localhost_access_log.2015-01-27.txt
-rw-r--r-- 1 root   root   1827 ene 28 21:05 localhost_access_log.2015-01-28.txt
-rw-r--r-- 1 root   root   3901 ene 29 14:22 localhost_access_log.2015-01-29.txt
```

Figura 126: Archivos de log de Apache Tomcat en Eclipse

Ahora tenemos la dirección IP del atacante que utilizó un browser común y la dirección IP del atacante que utilizó Tor Browser.

```
190.182.157.23 - - [29/Jan/2015:13:51:46 -0300] "GET /Login/ HTTP/1.1" 200 7371
190.182.157.23 - - [29/Jan/2015:13:51:57 -0300] "GET /Login/index.jsp?user=manager&pass=%27+OR+%27%27%3D%27 HTTP/1.1" 200 259
50.7.159.136 - - [29/Jan/2015:14:22:00 -0300] "GET /Login/ HTTP/1.1" 200 7371
50.7.159.136 - - [29/Jan/2015:14:22:50 -0300] "GET /Login/index.jsp?user=manager&pass=%27+OR+%27%27%3D%27 HTTP/1.1" 200 259
```

Figura 127: Contenido del archivo de log registrando los ataques SQLi

Podemos ver que obtenemos la ubicación verdadera que publica el ISP del atacante:

190.182.157.23

Hostname	23.157.182.190.unassigned.ridsa.com.ar
Network	AS27983 Red Intercable Digital S.A.
City	Buenos Aires, Distrito Federal, Argentina
Latitude/Longitude	-34.6033,-58.3816

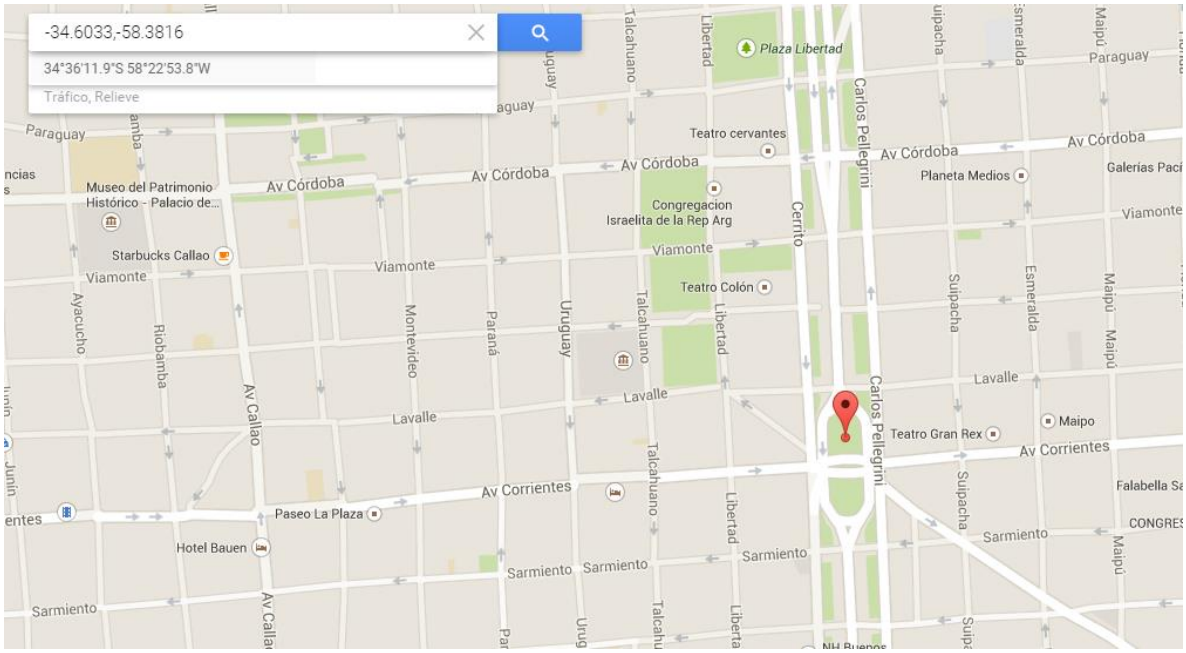


Figura 128: Ubicación del atacante público

Pero si buscamos la ubicación de la dirección IP de Tor Browser veremos que aparece en cualquier parte del mundo y no es posible rastrear al atacante.

50.7.159.136

Hostname	de4.kiwinet.eu
Network	AS6461 Abovenet Communications, Inc
City	Frankfurt, Hessen, Germany
Latitude/Longitude	50.1167,8.6833

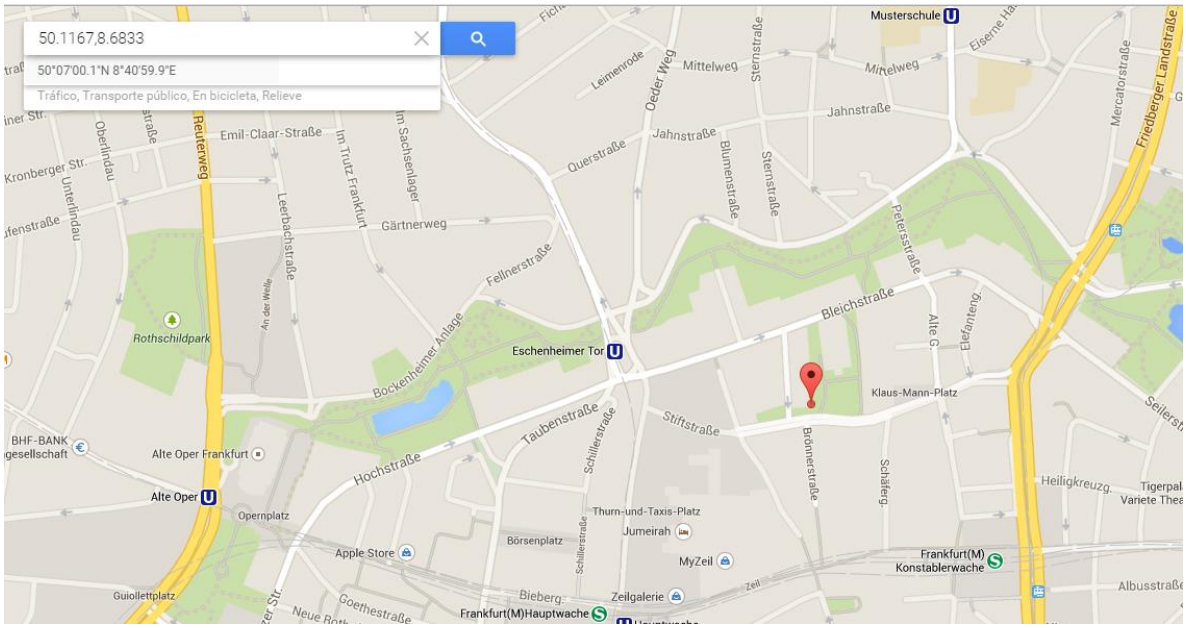


Figura 129: Ubicación del atacante anónimo

17.13. Aplicación del script iptables_tor_black_list.sh

Ejecutamos por primera vez el script sin una base de datos previa y obtenemos los siguientes resultados:

```
[root@192 ~]# ./iptables_tor_black_list.sh
--2015-01-29 16:35:31--  http://torstatus.blutmagie.de/ip_list_all.php/Tor_ip_li
st_ALL.csv
Resolviendo torstatus.blutmagie.de (torstatus.blutmagie.de)... 192.251.226.204,
2a02:cbf0:10:101::1:8324
Conectando con torstatus.blutmagie.de (torstatus.blutmagie.de)[192.251.226.204]:
80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: no especificado [application/force-download]
Grabando a: “Tor/Tor_ip_list_ALL”

    [       <=>                                 ] 99.310      56,0KB/s   en 1,7s

2015-01-29 16:35:33 (56,0 KB/s) - “Tor/Tor_ip_list_ALL” guardado [99310]

Nuevos nodos bloqueados: 7222
Total nodos bloqueados: 7222
```

Figura 130: Ejecución del script iptables_tor_black_list.sh en el border router

Por supuesto, la dirección IP de la red Tor, **50.7.159.136**, encontrada en el log del servidor web, se puede hallar en el archivo **Tor/Tor_black_list**:

```
[root@192 ~]# cat Tor/Tor_black_list | grep 50.7.159.136
50.7.159.136
```

Figura 131: Verificación de la dirección IP Tor activa en ese momento desde donde se realizó el ataque

Algo interesante que se pudo observar es que al no poder ingresar al sitio deseado con la IP que Tor nos asignó originalmente, se empieza a rotar la identidad, es decir la dirección IP, para intentar ingresar al sitio deseado. Por este motivo es muy importante que nuestra protección en el router sea dinámica, bloqueando a todos los nodos Tor activos en el momento del ataque.

Una vez que se realizan varios intentos de acceso podemos ver que resulta efectiva nuestra medida de seguridad:

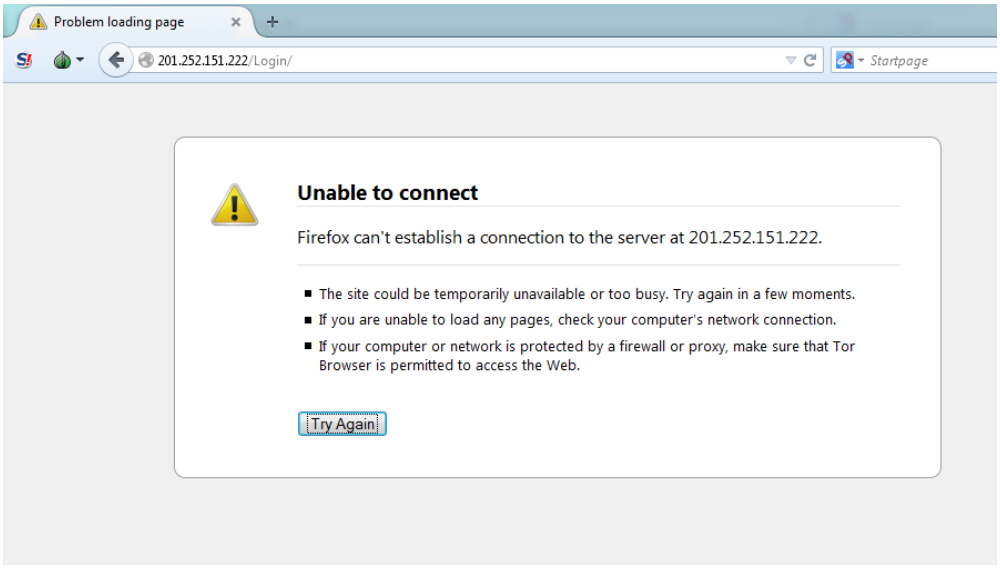


Figura 132: Imposibilidad de acceso desde Tor Browser luego de ejecutar el script iptables_tor_black_list.sh

17. CONCLUSIONES

Durante este proyecto se adquirieron muchos conocimientos sobre seguridad, redes anónimas, infraestructuras críticas, etc. pero también se puso a prueba y se refinó la capacidad de investigación personal. Por otro lado, se trabajó con tecnologías que no se habían visto en profundidad en los años de cursado, como es el caso de las bases de datos orientadas a documentos.

Una clara dificultad y limitación fue la de no poder acceder a información real acerca de cómo está conformada la infraestructura crítica del sistema de radarización de tráfico aéreo. Esto produjo idas y vueltas durante el desarrollo del proyecto. Por este motivo se retrasaron varias tareas hasta que se decidió hacer una hipotética estructura de red que se podría asemejar a la real y trabajar con ella.

Los resultados alcanzados al finalizar este trabajo de investigación fueron los siguientes:

- Se adquirieron conceptos sobre infraestructuras críticas y redes anónimas.
- Se obtuvieron datos analíticos sobre la actividad de los nodos de TOR en un período de tiempo determinado.
- Se desarrollaron métodos para mitigar ataques montados desde TOR.
- Se publicó un paper en el CoNaISII sobre el tema tratado, la investigación, el análisis y las pruebas realizadas hasta el momento.
- Se conformó exitosamente la infraestructura crítica planteada.
- Se pudo observar un comportamiento de rotación de identidad por parte de TOR para intentar eludir los métodos de mitigación desarrollados.
- Se verificó la efectividad de los métodos de mitigación desarrollados.

El border router que representa la salida a Internet desde la infraestructura crítica puede trabajar cargando las reglas de toda la base de datos de nodos Tor (aproximadamente 80.000, hasta el momento) o se puede optar por bloquear solamente los nodos activos en un determinado momento. Esta decisión depende del nivel de rendimiento que se precise o de los recursos de procesamiento con que se cuenta.

En cuanto a la solución propuesta para los bridges Tor, es a través de una recolección lenta y puede servir como histórico y tener una lista negra de los nodos bridge. De todas formas esta medida es para que los usuarios internos no se puedan conectar a la red Tor. Se pueden tomar otras medidas de seguridad como impedir descargas de internet a los usuarios o impedir el acceso a internet, no dejarlos colocar periféricos en las estaciones de trabajo, tales como pen drives, bloquear la ejecución de determinados programas, no dar permisos de instalación de programas en las estaciones de los usuarios, entre otras.

18. REFERENCIAS Y BIBLIOGRAFÍA

[1] Xiao Wang, Jinqiao Shi, Binxing Fang and Li Guo: “An Empirical Analysis of Family in the Tor Network”, IEEE ICC 2013 - Communication and Information Systems Security Symposium.

[2] Muhammad AliyuSulaiman and Sami Zhioua: “Attacking Tor through Unpopular Ports”, 2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops.

[3] AbdelberiChaabane, Pere Manils, Mohamed Ali Kaafar: “Digging into Anonymous Traffic: a deep analysis of the Tor anonymizing network”, 2010 Fourth International Conference on Network and System Security.

[4] Yang ZHANG: “Effective Attacks in the Tor Authentication Protocol”, 2009 Third International Conference on Network and System Security. Conference on Software Engineering (ICSE'03), Portland, OR, May 3-10 2003, pp. 125-137.

[5] Douglas Kelly, Richard Raines, Rusty Baldwin, Michael Grimaila, and Barry Mullins: “Exploring Extant and Emerging Issues in Anonymous Networks: A Taxonomy and Survey of Protocols and Metrics”, IEEE Communications Surveys & Tutorials, VOL. 14, NO. 2, Second Quarter 2012.

[6] Peipeng Liu, Jinqiao Shi, Lihong Wang, Xiao Wang, Qingfeng Tan: “IX-Level Adversaries on Entry- and Exit-Transmission Paths in Tor Network”, 2013 IEEE Eighth International Conference on Networking, Architecture and Storage.

[7] StjepanGroš, Marko Salki´c, Ivan Šipka: “Protecting TOR exit nodes from abuse”, MIPRO 2010, May 24-28, 2010, Opatija, Croatia.

[8] Eric Chan-Tin, Jiyoung Shin and Jiangmin Yu: “Revisiting Circuit Clogging Attacks on Tor”, 2013 International Conference on Availability, Reliability and Security.

[9] Florian Tschorschand BjörnScheuermann: “Tor is Unfair – and What to Do About It”, 36th Annual IEEE Conference on Local Computer Networks.

[10] Nicholas A. Fraser, Douglas J. Kelly, Richard A. Raines, Rusty O. Baldwin, and Barry E. Mullins: “Using Client Puzzles to Mitigate Distributed Denial of Service Attacks in the Tor Anonymous Routing Environment”, IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

[11] The Tor Project, Inc, “Tor project”: ”http://www.torproject.org

[12] Redesanónimas, I2P, FreeNet, TOR: “www.wikipedia.org".

[13] Comparación redes anónimas: “http://thehackerway.com/2012/02/08/preservando-el-anonimato-y-extendiendo-su-uso-comparacion-de-redes-anonimas-y-conclusiones-finales-parte-xxxxii/”

[14] Funcionamiento interno de TOR: "http://thehackerway.com/2011/11/25/preservando-el-anonimato-y-extendiendo-su-uso-funcionamiento-de-los-directorios-autoritativos-y-de-cache-en-tor-parte-xx/"

[15] Configuración ssmtp: "http://tuxedlinux.wordpress.com/2008/02/14/ssmtp-en-gmail/"

[16] Configuración fetchmail Gmail: "http://www.axllent.org/docs/view/gmail-pop3-with-fetchmail/"

[17] Configuración fetchmail Linux: "http://www.escomposlinux.org/Faq/FAQ_Linux_V2.0.2_html/FAQ_Linux_V2.0.2-100.html"

[18] Comandos Bash: "http://www.linux-es.org/node/238"