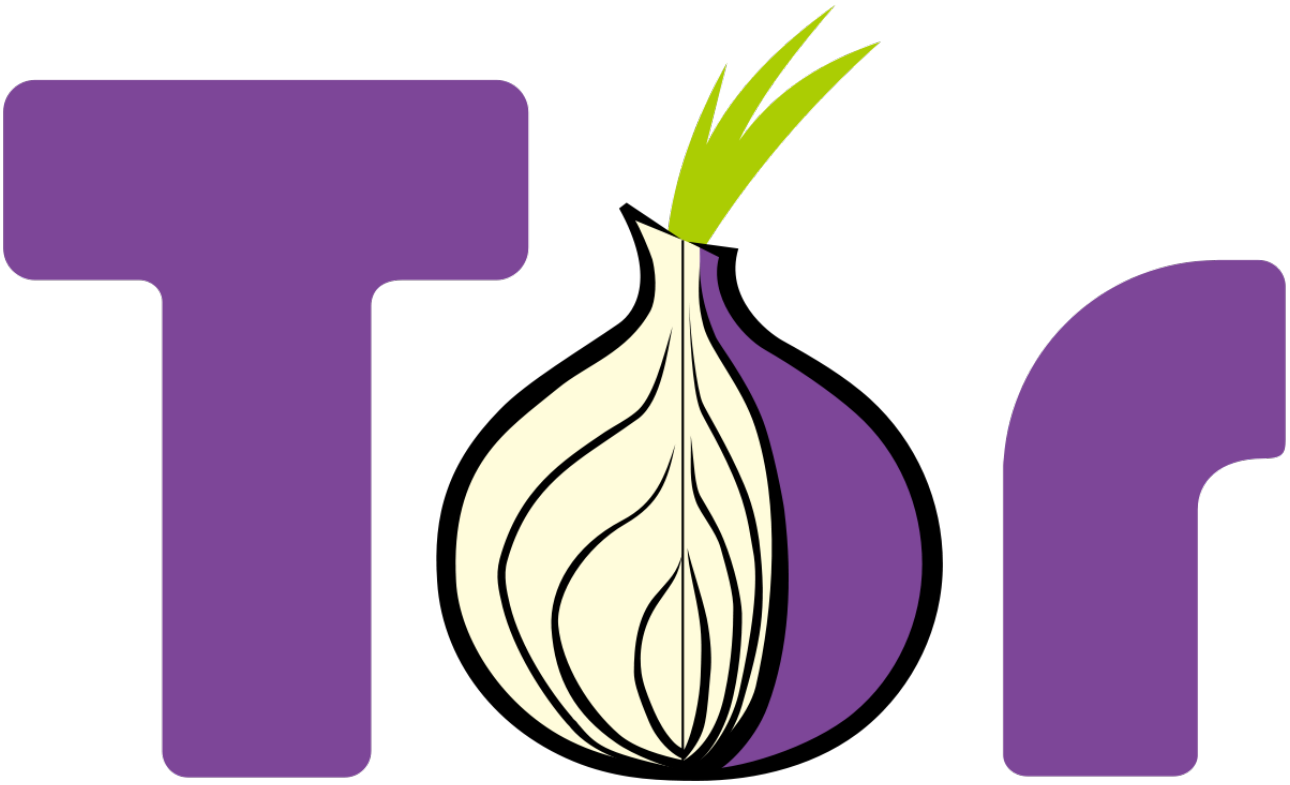
Creating a Tor Service



Índice:

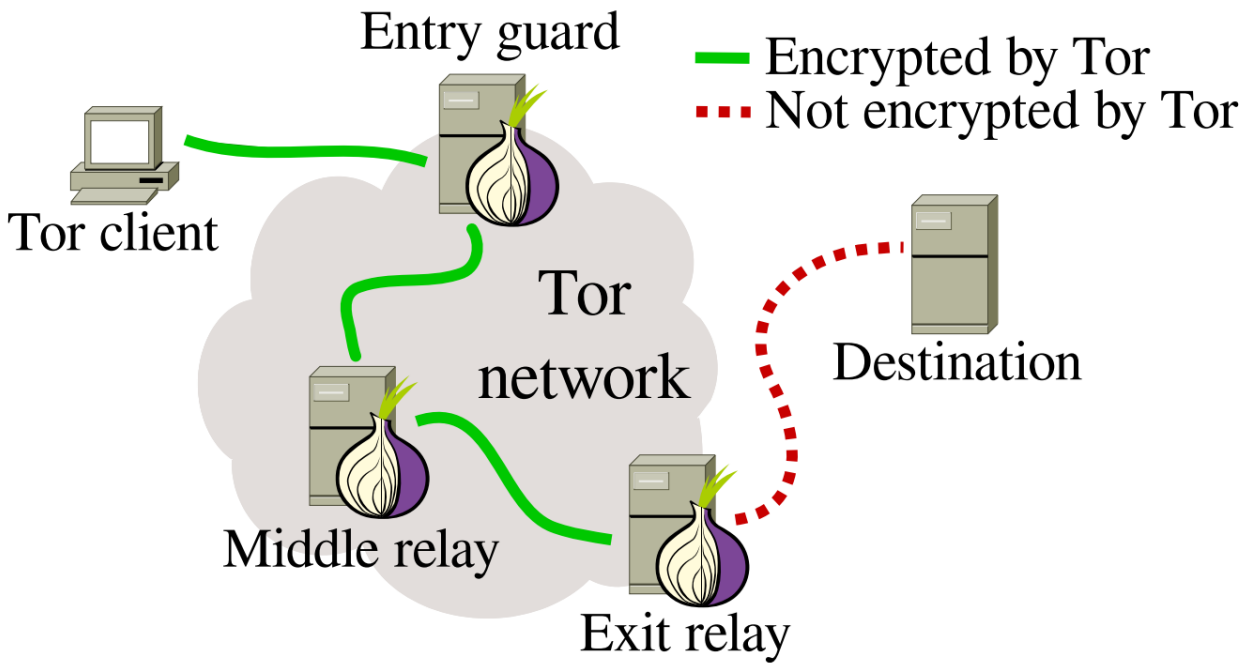
¿Qué es Tor?

Tor es un proyecto que tiene como objetivo desarrollar una red de comunicaciones distribuida y anónima. Consiste en una serie de servidores a los que los usuarios se conectan de forma que queda oculta su verdadera dirección IP, logrando así privacidad y evitando censuras o análisis en la red.

Los servidores a los que se conectan los usuarios están distribuidos en todo el mundo y son de tres tipos: de entrada, intermedio y de salida.

Estos servidores pueden ser mantenidos por universidades, organizaciones o casas particulares. A la conexión entre el usuario, generalmente tres de estos nodos y el servidor final que provee el servicio web se le llama **circuito**.

El modo en el que se establece el circuito es el siguiente.
El usuario se conecta a la red Tor (A través del Tor Browser por ejemplo) y selecciona un nodo de salida, establece la conexión cifrada con el nodo de salida y después selecciona un nodo intermedio. Tras establecer una nueva conexión cifrada con el nodo intermedio selecciona un nodo de entrada y establece una nueva conexión cifrada. De esta forma el nodo inicial tiene tres capas de cifrado, el nodo intermedio dos y el nodo final una, de manera que el usuario final queda totalmente anónimo en la conexión.



Cómo funciona tener un servicio en Tor

En la red Tor no existe un servicio DNS para que a la hora de contactar un servicio (hidden service) se pueda traducir un dominio en una IP, ya que eso supondría revelar la identidad de quien tiene el servicio. En su lugar existe una especie de anillo o directorio de servidores que es el encargado de hacer dicha función.
Cada servidor del anillo tiene un descriptor del servicio, llamado **HSDir** (Hidden Service Descriptor) que contiene datos sobre el servicio, generalmente valores de tiempo y claves (por ejemplo el tiempo que lleva activo, la clave cifrada 'diaria' de seguridad, llaves públicas, parámetros de consenso para establecer la conexión, etc).

Para evitar que un atacante pueda localizar un servicio a traveś del monitoreo de HSDir, éstos cambian de localización con el tiempo. Este tiempo se establece en la variable '**hsdir-interval**' establecida por consenso entre 30 minutos o 14400 (30 días), por defecto es 1 día.

Cuándo publicar un hidden service descriptor

Los hidden services publican periódicamente su descriptor en los correspondientes HSDir, cada vez que esto ocurre se establece de forma aleatoria un tiempo por el que es válido, entre 60 o 120 minutos. Si se sobrepasa ese tiempo, deberán ser publicados nuevamente.

Y si coinciden dos descriptores

Los hidden services tienen que subir multiples descriptores para que sean alcanzables por clientes con consensos más o menos actualizados que ellos. Además deben hacerlo al principio de cada periodo de tiempo, de manera que al siguiente reinicio estén actualizados y listos para usarse.
De hecho los servicios deben seguir subiendo su descriptor antiguo al HSDir para el caso en el que los clientes no tengan el consenso actualizado, puedan seguir accediendo al servicio con el antiguo.

Es por ello que en el HSDir se mantienen dos descriptores, el más nuevo y el anterior. Los clientes por otro lado no tienen constancia de que haya dos descriptores y acceden siempre al del tiempo y clave compartida actual.

Dónde publicar un hidden service descriptor

El conjunto de servidores donde se publican los HSDir se denomina anillo porque el proceso de selección es básicamente un bucle que los recorre en esa forma.

El sitio en el que se guarda el descriptor se determina por los siguientes parámetros: '**hsdir_n_replicas**', '**hsdir_spread_fetch**' y '**hsdir_spread_store**'. Que básicamente son números enteros establecidos por consenso.

De esa forma, para conocer el sitio en el que se guardará el descriptor en un determinado periodo de tiempo, el encargado de subir el descriptor o el de descargarlo hace el siguiente cálculo:

for replicanum in 1...**hsdir_n_replicas**:
hs_index(replicanum) = H("store-at-idx" |
blinded_public_key |
INT_8(replicanum) |
INT_8(period_length) |
INT_8(period_num))

Donde podemos ver que se trata de un bucle que recorre desde 1 hasta el número de réplicas de **hsdir**, el index o la posición del hidden service será el resultado de "**store-at-idx**", la clave pública, el valor de réplica y la longitud y valor del periodo (mencionado anteriormente).

De manera similar se escoje al nodo y al índice del directorio de ese nodo.

Para seleccionar un HSDir, los clientes eligen de forma aleatoria a partir del valor **'hsdir_spread_fetch'.**

Direcciones .onion

Las direcciones .onion en el HSDir son de mayor longitud que a las que accedemos desde el navegador, ya que se concatena la clave pública, el checksum y la versión, además se hace un cifrado en **base32**.

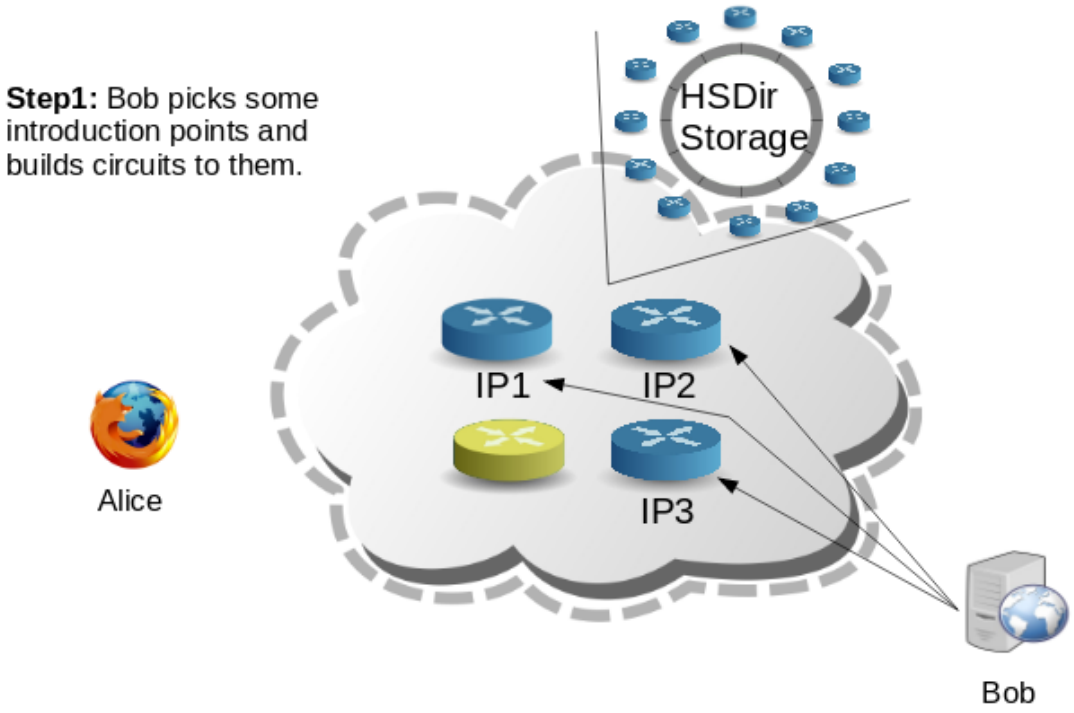
onion_address = base32(PUBKEY | CHECKSUM | VERSION) + ".onion"
CHECKSUM = H(".onion checksum" | PUBKEY | VERSION)

Ejemplos de direcciones son los siguientes:

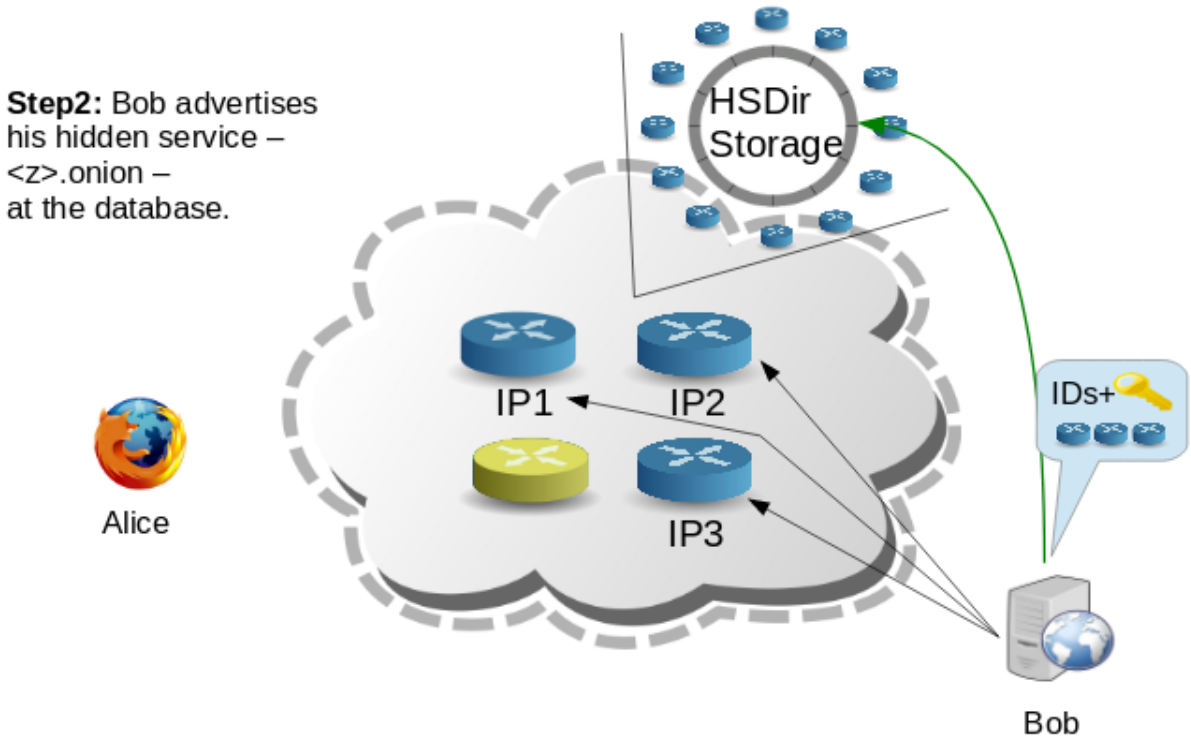
pg6mmjiyjmcrsslvykfwnntlaru7p5svn6y2ymmju6nubxndf4pscryd.onion
sp3k262uwy4r2k3ycr5awluarykdpag6a7y33jxop4cs2lu5uz5sseqd.onion

Conexión entre el cliente y el hidden service

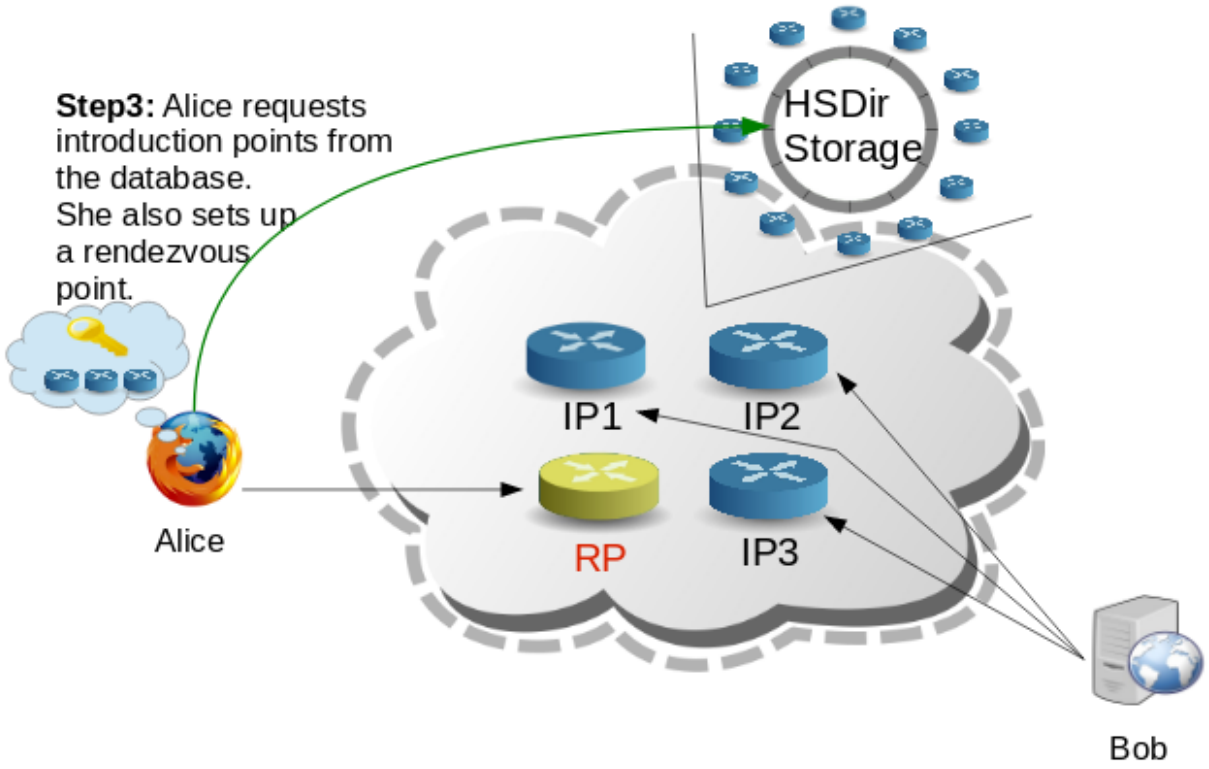
Ya hemos explicado las partes implicadas en la conexión entre un cliente y un hidden service, ahora vamos a explicar el proceso.



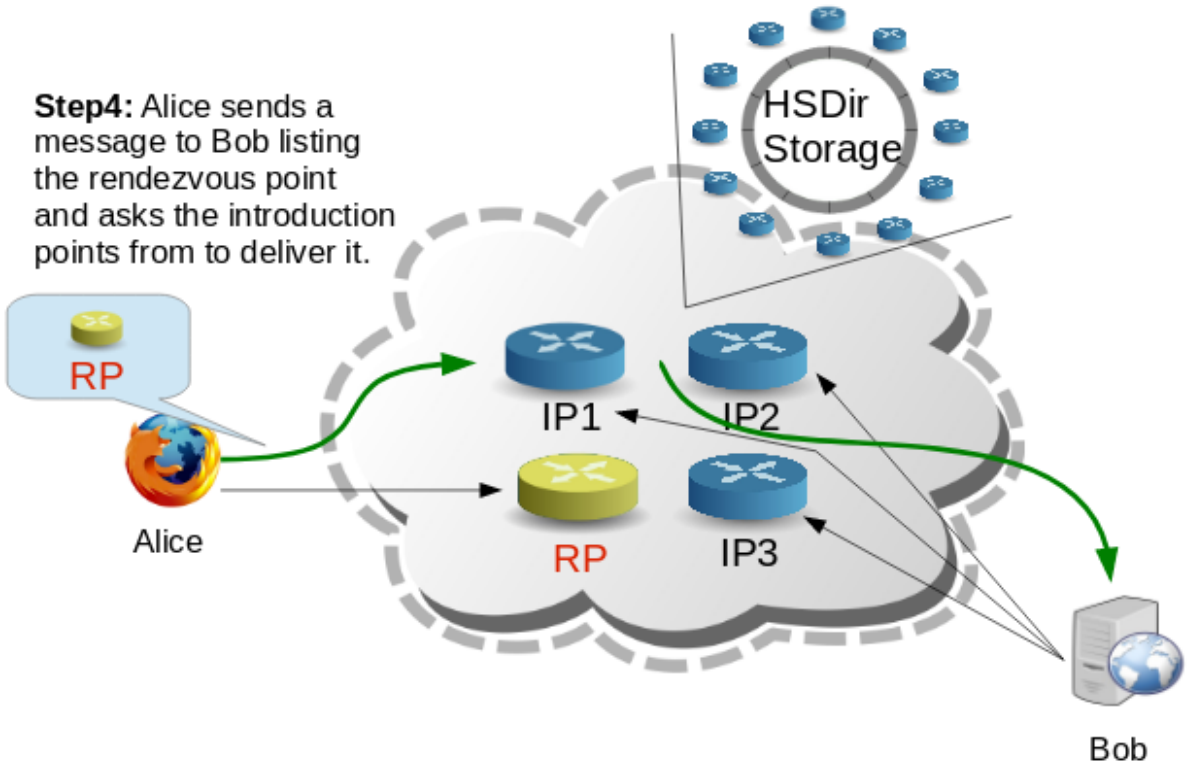
1. En el primer paso, el hidden service escoje una serie de **puntos de introducción**, estos servidores no son para establecer la comunicación sino para ser una especie de intermediarios entre los clientes y el hidden service.



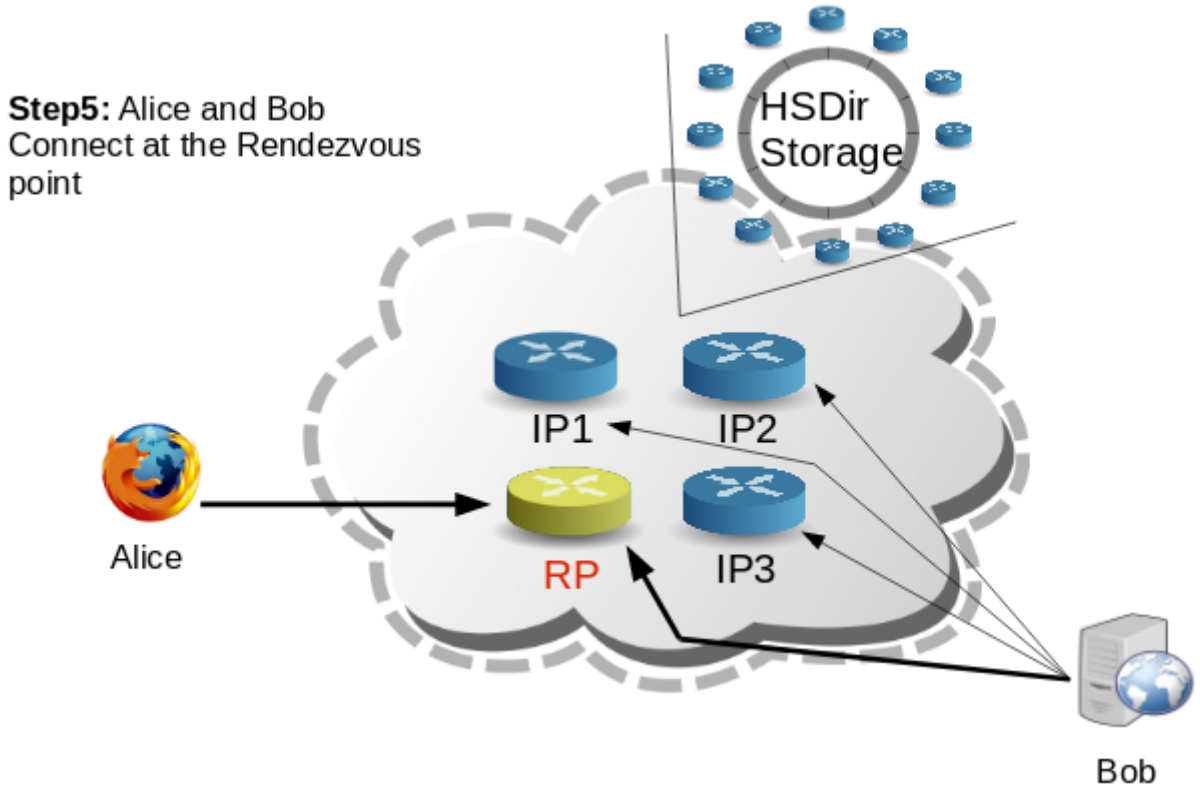
2. En el segundo paso el hidden service sube toda la información (IDs, claves, puntos de introducción…) al anillo de servidores HSDir.



3. El cliente solicita la información del servicio (claves y puntos de introducción) del anillo HSDir y además establece un **punto de reunión** que es el que se encargará de la comunicación.



4. Como el cliente conoce los puntos de introducción, establece una conexión con uno de ellos y le comunica al hidden service el **punto de reunión** que ha elegido previamente.



5. El cliente y el hidden service se conectan en el **punto de reunión** establecido.

Instalación

Explicado por pasos:

Nginx

1. Instalar Nginx

```
root@valkyrie:~# apt-get install nginx
```

2. Hacer una copia de seguridad de la configuración actual

```
root@valkyrie:~# cp /etc/nginx/sites-available/default
                   /etc/nginx/sites-available/default.old
```

3. Crear una nueva configuración

```
root@valkyrie:~# rm /etc/nginx/sites-available/default
root@valkyrie:~# vim /etc/nginx/sites-available/default
```


Configuración básica para que funcione:

```
server {
        listen 127.0.0.1:8080 default_server;
        server_name localhost;
        root /usr/share/nginx/html;
        index index.html;
        location / {
                allow 127.0.0.1;
                deny all;
        }
}
```

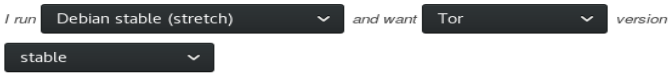
Solo permitimos **localhost** para que sea la red Tor la encargada de distribuir el servicio y no podamos ser accesibles por otro medio.

Nuestra web estará alojada en **/usr/share/nginx/html**

Tor

1. Seleccionar la versión correspondiente

```
root@valkyrie:~# lsb_release -a
No LSB modules are available.
Distributor ID: Kali
Description:    Kali GNU/Linux Rolling
Release:        kali-rolling
Codename:       kali-rolling
```



2. Añadir la fuente a apt

```
root@valkyrie:~# vim /etc/apt/sources.list
```

```
deb https://deb.torproject.org/torproject.org stretch main
deb-src https://deb.torproject.org/torproject.org stretch main
```

3. Verificar claves

```
root@valkyrie:~# gpg --keyserver keys.gnupg.net --recv A3C4F0F979CAA22CDBA8F512EE8CBC9E886DDD89
gpg: keybox '/root/.gnupg/pubring.kbx' created
gpg: key EE8CBC9E886DDD89: 1 duplicate signature removed
gpg: key EE8CBC9E886DDD89: 78 signatures not checked due to missing keys
gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: key EE8CBC9E886DDD89: public key "deb.torproject.org archive signing key" imported
gpg: no ultimately trusted keys found
gpg: Total number processed: 1
gpg:               imported: 1
root@valkyrie:~# gpg --export A3C4F0F979CAA22CDBA8F512EE8CBC9E886DDD89 | sudo apt-key add -
OK
```


4. Instalar

```
root@valkyrie:~# apt install tor deb.torproject.org-keyring
```

5. Instalar también la versión comprimida .tar de la web

Configurar Tor

1. Editar la configuración de tor, establecemos como directorio del hidden service: **/var/lib/tor/hidden_service/**

```
root@valkyrie:~# vim /etc/tor/torrc
```

```
############### This section is just for location-hidden services ###

## Once you have configured a hidden service, you can look at the
## contents of the file ".../hidden_service/hostname" for the address
## to tell people.
##
## HiddenServicePort x y:z says to redirect requests on port x to the
## address y:z.

HiddenServiceDir /var/lib/tor/hidden_service/
HiddenServicePort 80 127.0.0.1:8080

#HiddenServiceDir /var/lib/tor/other_hidden_service/
#HiddenServicePort 80 127.0.0.1:80
#HiddenServicePort 22 127.0.0.1:22
```

2. Restart

```
root@valkyrie:~# service nginx restart
root@valkyrie:~# service tor restart
```


3. Sorpresa

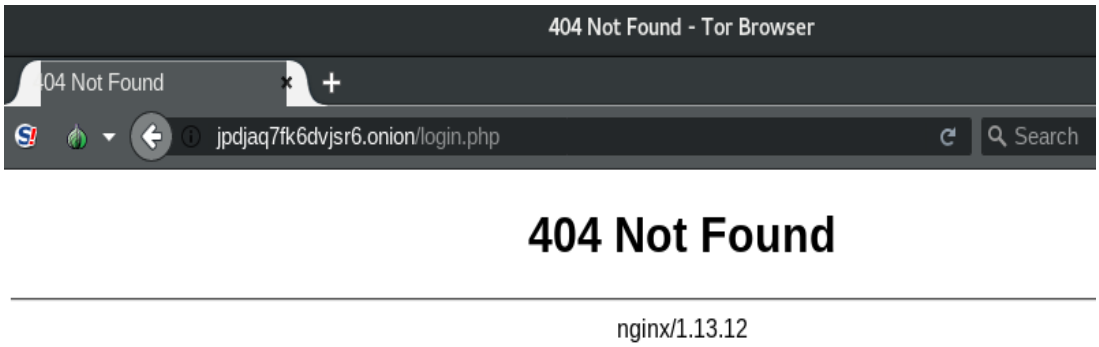
```
root@valkyrie:~# vim /var/lib/tor/hidden_service/
hostname     private_key
root@valkyrie:~# cat /var/lib/tor/hidden_service/hostname
jpdjaq7fk6dvjsr6.onion
```

En **/var/lib/tor/hidden_service/** se crean dos archivos, uno con la clave privada del servicio (no publicar ya que podrían hacerse pasar por nuestro servicio) y el dominio .onion del servicio en el archivo **hostname.**

Ya podemos acceder a nuestro index.html en /usr/share/nginx/html si ponemos ese dominio en el Tor Browser.

Hacer nuestro sitio seguro

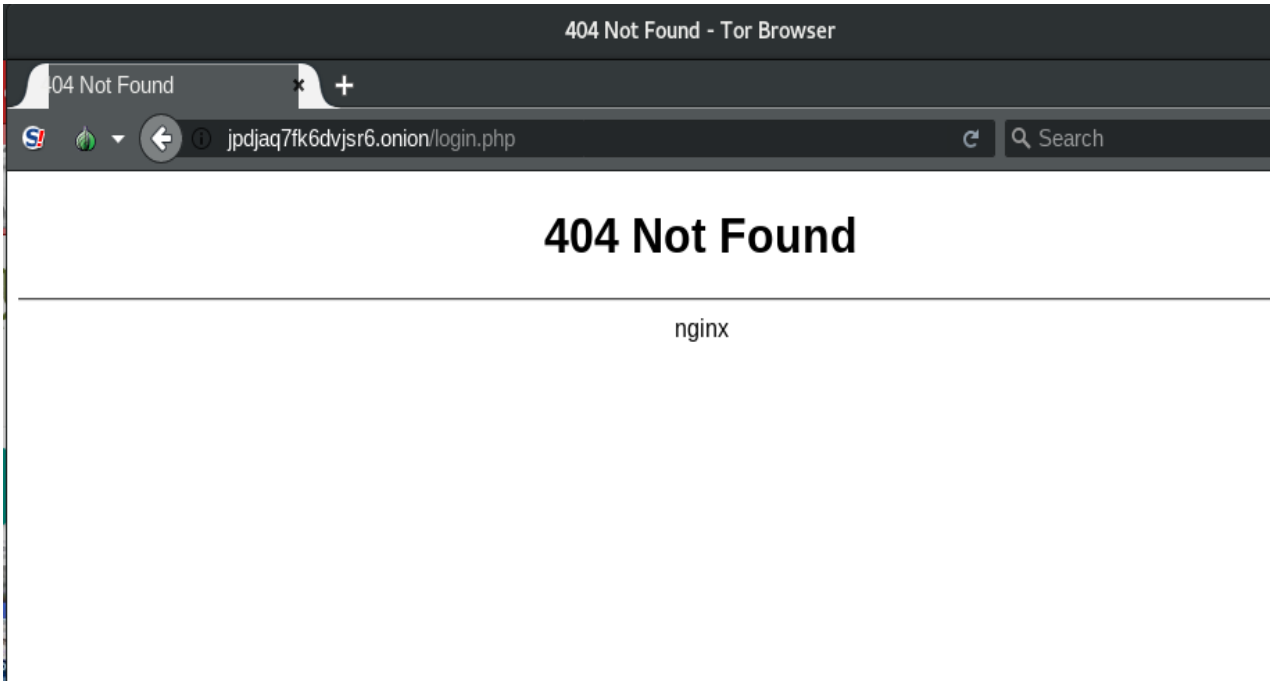
Si pedimos un recurso a nuestro servicio que no está disponible, por ejemplo una página 'login.php', debemos ocultar la versión de Nginx del mensaje de error de respuesta.



Esto se hace añadiendo: **server_tokens off;** a la configuración de Nginx.

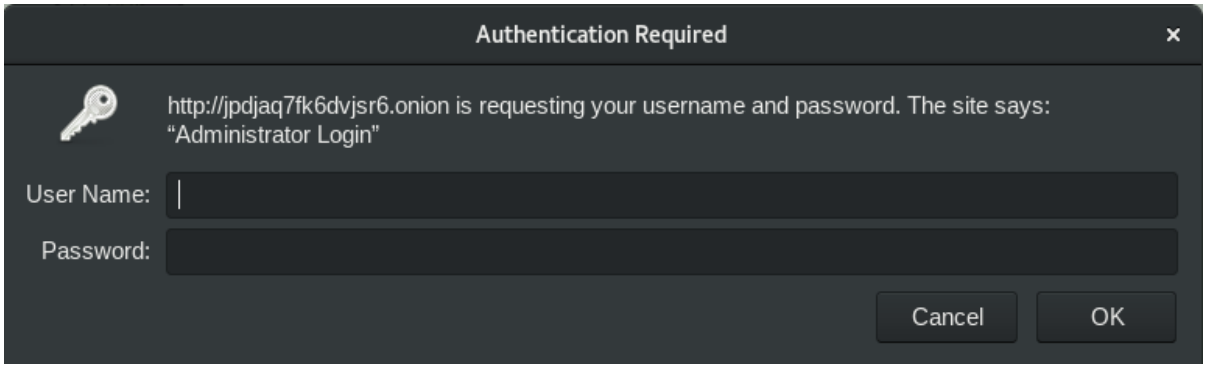
```
server {
        listen 127.0.0.1:8080 default_server;
        server_name localhost;
        root /usr/share/nginx/html;
        index index.html index.html;
        server_tokens off;
        location / {
                allow 127.0.0.1;
                deny all;
        }
}
```

Si lo comprobamos:



Añadir login a nuestro sitio

Si queremos añadir login:



Creamos una contraseña con perl y editamos el archivo **.htpasswd** para añadirla

```
root@valkyrie:~# perl -le 'print crypt("admin1", "1xzcq")'
1xlkpf6ZgGiVU
```

```
root@valkyrie:~# cat .htpasswd
admin1:1xlkpf6ZgGiVU
```

Lo añadimos a la configuración de Nginx, esta sería la **configuración final.**

```
root@valkyrie:~# cat /etc/nginx/sites-enabled/default

server {
        listen 127.0.0.1:8080 default_server;
        server_name localhost;
        root /usr/share/nginx/html;
        access_log /var/log/nginx/localhost.access.log;
        error_log /var/log/nginx/localhost.error.log;
        index index.html index.html;
        server_tokens off;
        auth_basic "Administrator Login";
        auth_basic_user_file /root/.htpasswd;
        location / {
                allow 127.0.0.1;
                deny all;
        }
}
```

Para el login, añadimos las líneas:

auth_basic “Administrator Login” # Puede ser un mensaje cualquiera

auth_basic_user_file /root/.htpasswd # Donde esta el usuario y contraseña

Además hemos añad[illegible] parámetros **acces_log** y **error_log** respectiva