Direcciones Efímeras en IPv6

Borja Marcos
borjam@sarenet.es

Gracias a la abundancia de direcciones que ofrece IPv6, deja de ser necesario recurrir a mecanismos de NAT. Pero este cambio trae consigo algunas consecuencias desde el punto de vista de la privacidad y la seguridad,. En los RFC 3041 y 4941 se ofrece una solución a uno de estos problemas: evitar el seguimiento a largo plazo de un nodo cuando se conecta en distintas ubicaciones. Sin embargo, la implementación actual este mecanismo aumenta la "superficie atacable" de un cliente. Por otro lado, la idea propuesta en estos RFC tiene el potencial de servir para mejorar aspectos de seguridad y privacidad.

En este artículo se propone una mejora en la implementación de las direcciones temporales, además de una generalización del mecanismo que, como novedad, se ofrecería como servicio a las aplicaciones. Este servicio sería especialmente útil, por ejemplo, para el mecanismo de "navegación privada" de los navegadores web o para servicios como el DNS recursivo.

Introducción

Eliminado el problema de la escasez de direcciones, en IPv6 el uso de NAT se hace innecesario. Sin embargo, esto trae consigo un importante cambio de condiciones con un importante impacto tanto desde el punto de vista de la seguridad como el de la privacidad.

NAT y cortafuegos

En primer lugar y aunque quede feo el decirlo, el carácter asimétrico del NAT hace que toda conexión a Internet en la que se utiliza tenga por defecto una protección contra las conexiones entrantes. Esta es la principal función de un cortafuegos y, en principio, la más simple. La mayor parte de los cortafuegos, de hecho, están configurados como simples NAT y podrían sustituirse por simples *routers*.

La existencia de NAT convierte de hecho en "clientes" a los nodos situados detrás del mismo. Con IPv6, eliminado el NAT, desaparece esa asimetría devolviendo la red a su carácter original de igual a igual. Y esto trae consigo un primer problema. Mientras en

un acceso NAT es imprescindible autorizar de manera explícita una conexión entrante, en un acceso sin NAT es necesario instalar un filtro que deniegue explícitamente las conexiones entrantes. Esto hace que el impacto de un error de configuración sea radicalmente diferente. En el caso de un NAT, una mala configuración puede hacer que se dirija un acceso al servicio equivocado o bien que el acceso que queremos configurar simplemente no funcione. En ningún caso puede autorizar un acceso indiscriminado a todos los servicios presentes en una red interna porque es matemáticamente imposible.

En una red sin NAT, en ausencia de un mecanismo que deniegue las conexiones entrantes el *router* dejará pasar todos los paquetes entrantes. Tanto si se emplea un *router* sin filtros/listas de acceso como si éstos se configuran incorrectamente, todos los servicios internos estarán accesibles.

NAT y confusión de direcciones

Otra propiedad útil del NAT, esta vez desde el punto de vista de la privacidad, es que dificulta la identificación de usuarios individuales que emplean el mismo acceso. Los operadores de servicios web, por ejemplo, se ven obligados a apoyarse en las *cookies* para llevar a cabo el seguimiento de un nodo a través de distintas sesiones.

Los navegadores aprovechan de hecho esta propiedad del NAT en sus mecanismos de "navegación privada", consistentes en crear ventanas que no están asociadas al caché de *cookies* del navegador. De este modo, una sesión abierta desde una ventana de "navegación privada" parecería en principio originada por un usuario nuevo al crearse una nueva *cookie*. Con el usuario detrás de un NAT no es posible en principio saber si ambas conexiones se han establecido desde el mismo nodo (lo que sugiere que se trata del mismo usuario). En un acceso IPv6 sin NAT, sin embargo, las ventanas "privadas" abiertas desde un mismo nodo tendrán *cookies* distintas pero la misma dirección IPv6, mientras las sesiones abiertas desde distintos nodos se distinguirían también por tener direcciones IPv6 distintas.

La confusión desaparece y, aunque no se pueda hablar en términos absolutos, sí es evidente que se puede saber con mucha más certeza si dos sesiones "privadas" pertenecen al mismo usuario o no.

Clientes que son servidores

Tendemos a considerar los sistemas operativos de escritorio como clientes. Pero de hecho la línea que separaría un cliente de un servidor desapareció en cuanto se generalizó el uso de sistemas operativos multiusuario y multitarea.

Es muy frecuente tener distintos servicios corriendo en las máquinas, incluso en ordenadores portátiles. Desde servicios de administración remota como VNC o SSH a servicios para compartir impresoras o archivos, servicios de *streaming*, etc. Muchas veces el usuario no es consciente de ello y, de hecho, el control de estos servicios constituye un blanco móvil. Con sucesivas actualizaciones de los sistemas aparecen servicios nuevos, otros des-

aparecen por obsoletos y no pocas veces una actualización ha activado un servicio que el usuario había desactivado previamente.

En muchos casos permitir el acceso a estos servicios es una mala idea por motivos obvios: la clave de acceso que resulta adecuada en una red doméstica (si es que exite) puede ser un suicidio en una red ajena. Y especialmente peligrosa si hay un acceso sin restricciones desde cualquier punto de Internet. Es previsible, por otro lado, que algunos de estos servicios tengan algún problema de seguridad de vez en cuando. Una vez más, el impacto es mucho mayor si están accesibles *urbi et orbi*. Cada uno de estos servicios aumenta de hecho la *superficie atacable*.

Pero hay otro problema que se suele considerar con menos frecuencia: el impacto de dichos servicios sobre la privacidad. El mejor ejemplo es quizás ssh, que al recibir una conexión presenta una clave pública. Supongamos que un mismo ordenador se conecta a lo largo del día desde un aeropuerto, una estación de tren, un hotel y un cibercafé. Supongamos además que estos accesos son modernos, disponen de IPv6, y hagamos la suposición, no descabellada, de que no disponen de un cortafuegos funcional. Imaginemos que los propietarios de un servicio de *banners* publicitarios poco escrupuloso, sabiendo que es posible que haya un ssh activo, intentan una conexión al ssh de la dirección IP de los visitantes a su servicio web. Y, sorpresa, se encuentran la misma clave pública ssh en las cuatro ubicaciones a pesar de que la dirección IP es diferente.

Para obtener dicha clave pública (que es un dato que permite identificar un nodo con un grado más que notable de certeza) no es necesario autentificarse. Otros servicios pueden identificarse con nombres de nodo que contienen información personal ("Ordenador de Fulanito Mengánez"), ofrecer archivos para descarga como fotos, etc.

1. Direcciones temporales en IPv6

En esta sección se examina el mecanismo de direcciones temporales propuesto en los RFC3041 y RFC4941. Dado que el RFC4941 supone una mejora sobre el RFC3041, en lo sucesivo se sobreentiende que cualquier mención al RFC4941 se refiere asimismo al RFC3041.

1.1. SLAAC, RFC3041 y RFC4941

SLAAC permite que un nodo se asigne una dirección IPv6 sin necesidad de un coordinador, a diferencia de DHCP. Cada nodo se asigna una dirección IPv6 formada por el prefijo de red recibido del *router* y su propia dirección MAC. El hecho de que las direcciones MAC sean únicas impide que sea probable un conflicto de direcciones.

Este esquema tiene una ventaja muy importante: la simplicidad. Pero plantea un importante problema de privacidad. Un dispositivo nómada que se conecte en distintas ubicaciones en las que se emplee SLAAC obtendrá direcciones IPv6 en las que el sufijo

será reconocible y único. En estas condiciones resulta trivial el seguimiento de un usuario[1] a lo largo del tiempo.

En el RFC3041 se propone un mecanismo (mejorado en el RFC4941) de asignación de direcciones temporales aleatorias con un tiempo de vida limitado. Estas direcciones, creadas para ser empleadas al iniciar conexiones salientes, evitan el problema del seguimiento. Las direcciones generadas por el nodo al conectarse a una red no contienen su dirección MAC ni otro elemento identificable de manera visible. Asimismo, estas direcciones tienen un tiempo de vida limitado. Esto impide también un seguimiento a largo plazo en la misma ubicación. Por poner un ejemplo, el mismo dispositivo empleado durante varios días en la red de un hotel tendrá cada día una dirección temporal diferente.

1.2. RFC4941 e iluminación de blancos

El mecanismo propuesto se refiere a una situación muy bien definida. Como puede leerse en el punto 2 de la sección 3 (la negrita es mía):

> Create additional addresses based on a random interface identifier **for the purpose of initiating outgoing sessions**. These "random" or temporary addresses would be used for a short period of time (hours to days) and would then be deprecated. Deprecated address can continue to be used for already established connections, but are not used to initiate new connections. New temporary addresses are generated periodically to replace temporary addresses that expire, with the exact time between address generation a matter of local policy.

Se especifica claramente que estas direcciones temporales se emplearán con el propósito de iniciar conexiones. Sin embargo, el RFC **no** dice en ninguna parte que se empleen **de manera exclusiva** para iniciar conexiones.

Excepto en el caso de las conexiones iniciadas por el nodo en cuestión, las direcciones IPv6 temporales son direcciones válidas a todos los efectos. En particular, cualquier programa que esté escuchando en la dirección `in6addr_any` (la dirección IPv6 wildcard, equivalente a la clásica `INADDR_ANY` en IPv4) escuchará también en las direcciones temporales.

Imaginemos que un portátil se conecta a Internet primero en un aeropuerto y después en un hotel, ninguno de los cuales tienen un cortafuegos[2]. En cada uno de los casos obtendrá una dirección temporal diferente. Supongamos que tiene un servicio ssh con la configuración por defecto[3] escuchando en el puerto 22. El usuario visita unas cuantas páginas web que cargan *banners* publicitarios y, por tanto, el proveedor de publicidad registra visitas de dos direcciones IPv6 diferentes. Tratándose de un proveedor, digamos, cuestionable, por si acaso devuelve un intento de conexión al ssh de la dirección que acaba de acceder a su servicio. En ambos casos la conexión tiene éxito (el ssh está escuchando

[1] Asumiendo, claro está, una correspondencia entre usuario y dispositivo, correspondencia obvia en el caso de los dispositivos móviles.

[2] O bien está incorrectamente configurado, que viene a ser lo mismo

[3] ListenAddress 0.0.0.0
ListenAddress ::

en ::) y obtiene una clave pública que ¡coincide en ambos casos!. **Acaba de determinar con éxito que las dos direcciones IP sin relación aparente corresponden de hecho al mismo dispositivo**.

Al margen de los problemas de privacidad, esto plantea también un problema de seguridad. Se ha dicho muchas veces que el uso de cortafuegos en IPv6 puede no ser tan importante debido a que barrer una red IPv6 exige un esfuerzo considerable. Sin embargo, en el momento en el que un nodo inicia conexiones hacia el exterior, bien visitando algún servicio o, por ejemplo, participando en una red p2p, revela una dirección IPv6 en la cual estarán a la escucha los servicios locales.

Por hacer un símil bélico, en ausencia de cortafuegos un nodo IPv6 puede compararse a un barco en una guerra naval. Mientras en la guerra terrestre la seguridad depende sobre todo de una trinchera, en el mar una parte muy importante de la ventaja proviene de la incapacidad del enemigo para determinar nuestra posición. De hecho es muy importante mantener una disciplina de silencio de radio evitando emisiones que puedan ser captadas más allá del horizonte.

El aceptar por defecto conexiones en las direcciones temporales permite de hecho que un agresor emplee tácticas de *iluminación de blanco*. Supongamos que un agresor desea intentar acceder a un servicio del dispositivo de nuestro ejemplo. Bien mediante un mensaje de correo electrónico o por cualquier otro medio convence al usuario para que visite una página web determinada. En ese momento el agresor consigue sin recurrir a un barrido una dirección IPv6 en la que, recordemos, están a la escucha todos los servicios de su dispositivo.

1.3. Navegadores www y "ventanas privadas" con IPv6

Con el fin de dificultar el seguimiento de un usuario a través de distintos servicios, los navegadores www modernos ofrecen un mecanismo llamado "navegación privada". La idea es en principio muy simple: las ventanas privadas no comparten *cookies* entre sí y, por supuesto, no disponen de almacenamiento persistente de datos del servidor.

Este mecanismo de protección de la privacidad se beneficia en parte de la presencia de un NAT. En principio no es posible para el operador de un servicio WWW saber si dos visitas procedentes de la misma dirección IPv4 pero con *cookies* distintas proceden del mismo usuario o no. La presencia del NAT elimina la relación biunívoca entre dirección IP y dispositivo creando cierta confusión que resulta beneficiosa desde el punto de vista de la privacidad.

Sin embargo, con IPv6 desaparece dicha confusión. Un usuario que tenga abiertas varias ventanas de navegación privada accediendo al mismo servicio (por ejemplo, un servidor de publicidad al que se hace referencia desde varios sitios) será delatado por su dirección IPv6, que será la misma dentro del período de validez de las direcciones temporales (habitualmente 24 horas). El uso de IPv6 convierte la dirección IP en un elemento de seguimiento de usuarios a corto plazo. Con IPv4 la fiabilidad de la dirección es al menos dudosa.

2. Mejoras propuestas

En esta sección se detalla, por una parte, una mejora en el uso de las direcciones temporales. Por otro lado se propone un nuevo mecanismo que permite generar direcciones temporales de uso específico para cada aplicación/servicio/proceso sujeto además al control de la política de seguridad del sistema operativo.

2.1. Limitación de recepción de conexiones en las direcciones IPv6 temporales

La primera propuesta es un cambio en el uso de las direcciones temporales para recibir conexiones. Las direcciones temporales no deberán aceptar conexiones para servicios que estén escuchando en la dirección *wildcard*. Para aceptar una conexión (o un paquete UDP) dirigido a una dirección temporal será requisito indispensable que exista un socket que haya hecho un bind() a dicha dirección IPv6.

2.1.1. Implementación

La implementación de este cambio no es en principio complicada. Las direcciones temporales están identificadas como tales en las estructuras de datos del *kernel*. En el caso de TCP, por ejemplo, con un proceso escuchando en la dirección wildcard y el puerto N, la recepción de un paquete SYN dirigido al puerto 22 solamente se aceptará como inicio de conexión si la dirección IPv6 de destino no está marcada como dirección temporal. En caso de dirigirse a una dirección temporal, el paquete SYN se tratará exactamente de la misma forma que un paquete dirigido a un puerto en el cual no hay un proceso escuchando.

Es importante notar que la limitación se aplica única y exclusivamente al caso en el que el socket está escuchando en la dirección *wildcard*. Hay servicios que tras iniciar una conexión saliente requieren un segundo *socket* para recibir una conexión "de vuelta". En este caso la secuencia (con TCP) sería algo así:

```
s_inicial = socket(AF_INET6, SOCK_STREAM...)
s_escucha = socket(AF_INET6, SOCK_STREAM...)
connect(s_inicial, destino)
// Una vez conectado, podemos obtener la dirección local de la conexión
mi_ipv6 = getsockname(s_inicial...)
// y si necesitamos escuchar en la misma dirección, utilizarla
bind(s_escucha, mi_ipv6, puerto) // Al poner de forma explícita la dirección mi_ipv6
listen(s_escucha)
accept(s_escucha)
...
```

En la mayor parte de los casos este cambio de comportamiento de las direcciones temporales no exigirá cambios en el código. En un nodo con múltiples interfaces de red y,

por tanto, múltiples direcciones, es necesario determinar cuál es la dirección IP de origen de la conexión, que a su vez se determina en función de la ruta por la cual ha salido el paquete SYN. Este dato no se conoce hasta que, de hecho, se ha establecido una conexión con la llamada `connect()`.

2.1.2. Cambios en el API

Con el fin de poder controlar el uso de este nuevo mecanismo y limitar su posible impacto sobre aplicaciones antiguas, no obstante, se proponen varias opciones:

- Una variable de configuración que permita escoger el comportamiento antiguo (`net.inet6.listen_on_wildcard`). Por defecto su valor sería FALSE (o 0). En caso de ponerla a TRUE (o 1) se recuperaría el comportamiento anterior. La escucha en un *socket* con un `bind()` a la dirección *wildcard* incluiría también las direcciones IPv6 temporales. Dado que el impacto puede ser muy distinto dependiendo del protocolo en cuestión (o al menos del tipo de transporte del `socket`) puede ser recomendable definir variables de configuración por protocolo o transporte.
  - `net.inet6.stream.listen_on_wildcard`
  - `net.inet6.dgram.recv_on_wildcard`
  - `net.inet6.tcp.listen_on_wildcard`
  - `net.inet6.sctp.listen_on_wildcard`
  - `net.inet6.udp.recv_on_wildcard`
- Una opción que permita, pero de manera limitada al *socket* en cuestión, incluir las direcciones temporales a la hora de escuchar. La forma de uso sería algo similar a: `setsockopt(s, SO_LISTEN_ON_WILDCARD)`.

El ejemplo propuesto se refiere a TCP o, en general, a *sockets* de tipo *stream*. No obstante, en el caso de un transporte de datagramas el caso sería similar. Para recibir en una dirección IPv6 temporal concreta sería necesario hacer un `bind()` especificando dicha dirección o, en su defecto, emplear la opción SO_LISTEN_ON_WILDCARD (o, por consistencia con el funcionamiento de los *sockets* tipo datagrama, definir otra opción llamada `SO_RECV_ON_WILDCARD`.

2.2. Empleo de múltiples direcciones IPv6 efímeras

Como se ha mencionado en la introducción, el mecanismo de "navegación privada" queda muy limitado en cuanto se elimina la incertidumbre de la traducción de direcciones. El mecanismo propuesto en los RFC mencionados considera en general el uso de una dirección IP temporal que caduca pasado un tiempo, pero en un acceso IPv6 resulta trivial para el operador de un servicio web saber si dos ventanas "privadas" pertenecen al mismo nodo. La dirección IPv6 de origen de las conexiones será la misma.

Por otro lado, compartir una dirección IP de origen entre varias conexiones o aplicaciones puede ser un problema si no tenemos el mismo grado de confianza en los destinos. En un ataque de envenenamiento de cache de DNS, por ejemplo, el primer paso consiste en

averiguar la dirección de origen que emplea el servidor DNS para hacer las preguntas. El atacante *ilumina* el servidor DNS haciendo que uno de sus clientes haga una pregunta de un dominio que controla.

En IPv6, a diferencia de IPv4, el número de direcciones IPv6 disponible se puede considerar infinito a todos los efectos. Este hecho permite un cambio radical en el uso que hacemos de las mismas. Si en IPv4 tenemos a utilizar una dirección IPv4 por nodo, con IPv6 podemos permitirnos asignarlas por servicios u otros criterios e incluso crear y destruir direcciones efímeras de manera dinámica.

2.2.1. Propuesta de implementación

A la hora de implementar un mecanismo de este tipo conviene tener en cuenta los siguientes requisitos:

1. Hacerlo sencillo
2. Minimizar el impacto sobre las aplicaciones ya existentes
3. Sería deseable que un administrador de sistemas pudiera (dentro de lo razonable) imponer el uso de este mecanismo a aplicaciones sobre las que no puede hacer cambios.

El primer requisito nos obliga de antemano a descartar los mecanismos de manipulación de direcciones IP del sistema. Además de ser complicados de utilizar, son mecanismos privilegiados por motivos obvios.

Los requisitos 2 y 3 ya están de hecho cubiertos por el mecanismo propuesto en los RFC de las direcciones temporales. La implementación propuesta funciona en principio de la misma manera, esto es, una aplicación establecerá una conexión como siempre, empleando la llamada al sistema `connect()`. La diferencia está en el mecanismo de selección de dirección IP. Asumiendo que, como ocurre en la mayoría de los casos, un cliente no seleccione una dirección de origen específica mediante `bind()`, dejando al sistema operativo seleccionar la dirección de origen. ¿Cómo se selecciona la dirección de origen?

- En un sistema sin uso de direcciones temporales, se examina la dirección de destino del paquete inicial y se asigna como dirección de origen la dirección del interfaz correspondiente a la mejor ruta al destino.
- En un sistema con direcciones temporales se hace lo mismo que en el caso anterior. Pero se da preferencia como dirección de origen a la dirección temporal en vigor, si es que existe.
- Con direcciones efímeras deberemos llevar a cabo un paso más: generar una nueva dirección temporal que se empleará como origen para iniciar la conexión. Además, esta dirección temporal quedará marcada como "privada" y en ningún caso se asignará como origen de manera automática, aunque sí puede reutilizarse si se especifica en otro *socket* mediante `bind()`.

Aunque no debería tener efecto sobre las aplicaciones ya existentes, implementar esta funcionalidad sí implica hacer tres cambios de calado en la pila TCP/IP.

1. Definir un nuevo tipo de dirección IPv6. Se trata de la dirección IPv6 efímera **temporal** y **privada**.
2. Incluir en la pila un "cortocircuito" entre la capa de transporte y la capa de red al ser necesario que el inicio de una conexión genere una nueva dirección IPv6.
3. El tercer cambio, por supuesto, es un mecanismo de caducidad de las direcciones temporales privadas. Aunque las implementaciones modernas de IPv6 ya disponen de un mecanismo para suprimir las direcciones temporales, está orientado solamente a una caducidad temporal. En este caso sería deseable combinar un tiempo de caducidad mucho más corto con un contador de referencias. Cuando la dirección efímera lleve determinado tiempo sin utilizarse (esto es, sin estar asignada a ningún socket) deberá borrarse.

2.2.2. Cambios en el API

Desde el punto de vista de las aplicaciones el comportamiento es el mismo que el ya existente en el caso de las direcciones temporales. En todo caso es interesante añadir opciones para `setsockopt()` que permitan seleccionar o desativar el mecanismo de asignación de direcciones efímeras para un socket.

De la misma manera, una serie de variables de configuración (por ejemplo, mediante el mecanismo `sysctl` presente en FreeBSD, Linux, etc) permitirían al administrador configurar el comportamiento deseado (activarlo por defecto o no, utlizarlo o no, etc).

3. Aplicaciones de las direcciones efímeras

3.1. Generación de entropía para preguntas de DNS

El protocolo DNS ha sufrido de varias vulnerabilidades a lo largo de la historia. Al margen del despliegue de DNSSEC es imprescindible asegurar que el origen de una pregunta de DNS no sea predecible. Hoy día se recomienda configurar los servidores de manera que el puerto de origen varíe en un rango lo más amplio posible. Esto nos proporciona una incertidumbre de unos 16 bits a la hora de determinar el puerto de origen. Si empleamos un *pool* de direcciones IP como origen podremos arañar unos cuantos bits más, pero basándonos en un conjunto de direcciones muy limitado.

Con IPv6 esto cambia de manera radical. Si generamos de manera completamente aleatoria las direcciones partiendo de un rango /64 podemos conseguir una importante mejora en la incertidumbre. Dependiendo del número de direcciones que se mantengan vivas, además, conseguimos reducir enormemente el valor de una dirección IP de origen obtenida por un servidor DNS malicioso.

3.2. Aislamiento entre aplicaciones

Servicios como la telefonía IP y la navaegación web tienen características muy diferentes. Determinados servicios, además, pueden ser especialmente vulnerables ante ataques de

denegación de servicio por falsificación. Como ya se ha explicado antes, cuando el usuario se conecta con un servicio web (o con otro servicio, pero se trata del mejor ejemplo) está revelando su dirección IP. Si el usuario visita un servicio hostil y está utilizando la misma dirección IP para otros servicios al menos queda expuesto a un posible ataque de denegación de servicio.

A grandes rasgos este tipo de ataques de denegación de servicio puede dividirse en dos categorías:

- Ataques contra el cliente: Se trata del clásico envío de paquetes RST o errores de ICMP con el objeto de engañar al cliente para que cierre una conexión.
- Ataques indirectos contra servicios. Supongamos que un usuario móvil necesita conectar con un servicio en su oficina y su red dispone de un cortafuegos con algún mecanismo automático de lista negra para direcciones desde las que se detecte un posible incidente, por ejemplo un barrido de puertos. Sería trivial para el atacante, una vez conocida la dirección IP del usuario móvil, hacer un barrido de puertos (o algo similar) falsificando la dirección IP de origen. Este tipo de acción puede convertir un mecanismo de seguridad en un mecanismo de denegación de servicio.

Empleando direcciones efímeras diferentes se mantiene una separación entre ambas aplicaciones. Conocer la dirección efímera utilizada para navegar serviría para conocer el prefijo de red en el que el usuario esstá conectado, pero no para conocer la dirección utilizada en otro servicio, por ejemplo, la telefonía.

3.3. Privacidad en los navegadores

El mecanismo de navegación privada de los navegadores se basa únicamente en la gestión del almacén de *cookies* y otros datos proporcionados por los servidores web. Con un mecanismo de este tipo, se ofrece una nueva herramienta a los desarrolladores de navegadores y aplicaciones de privacidad. Algunas de las sugerencias de uso son:

- Aprovechar el mecanismo de direcciones efímeras en las ventanas privadas. De este modo las direcciones IPv6 vuelven a tener la falta de fiabilidad que tenían las IPv4 con NAT. Dos ventanas privadas abiertas en el mismo nodo no tendrán la misma dirección.
- Dentro de la carga de una página web, que con frecuencia incluye elementos de distintos origenes, podría resultar interesante que dicha carga se hiciera desde direcciones IPv6 diferentes. Se trataría de un complemento muy útil a las políticas actuales de limitación de carga de *cookies*.

Referencias

[1] Narten, Draves 2001. RFC3041 “Privacy Extensions for Stateless Address Autoconfiguration in IPv6”

[2] Narten, Draves, Krishan 2007. RFC 4941 - “Privacy Extensions for Stateless Address Autoconfiguration in IPv6”

[3] IETF 2008. “Comprehensive DNS Resolver Defenses Against Cache Poisoning draft-weaver-dnsext-comprehensive-resolver-00”