Si estamos buscando usar un ordenador/pc/servidor/máquina con Debian como proxy/zombie para realizar escaneos de tipo «idle» con nmap, tenemos que saber ciertas cosas:

En las distros GNU/Linux con kernel modernos ya no existe un simple ajuste por sysctl o parámetro de configuración que permita volver a la generación de identificadores IP (IP ID) puramente incremental de forma nativa. El comportamiento actual (que introduce aleatoriedad en el IP ID o usa secuencias "semi-aleatorias") es un mecanismo de seguridad para dificultar ataques de ciertas clase, como los propios idle-scans o el rastreo de conexiones. Por tanto, no basta con tocar un fichero en **/proc/sys/net/ipv4/** o hacer un cambio sencillo para forzar la secuencia incremental, ya que el código que gestiona el IP ID en Linux está en el propio kernel y no se controla mediante un sysctl. Entonces, a grandes rasgos, para transformar un Debian en un zombie hay dos posibles caminos:

1. Parchear y recompilar el kernel, para que la función que asigna el IP ID se comporte de la manera que queramos. Por ejemplo, que incremente un contador en lugar de usar randomización.
2. Reescribir el campo IP ID "en vivo" usando netfilter u otro mecanismo con algún módulo especial de iptables/nftables que reescriba el campo IP ID al vuelo.

Este segundo camino es más teórico que práctico. Podríamos interceptar el tráfico en espacio de usuario o con un módulo especial de iptables/nftables que **reescriba** el campo IP ID al vuelo. Sin embargo, en el árbol principal del código fuente de netfilter, no existe un módulo oficial que reescriba directamente el campo IP ID para todos los paquetes salientes. Podríamos aproximar algo customizado usando nfqueue, llevando paquetes a espacio de usuario, modificando la cabecera IP y reinyectándolos. Pero el rendimiento sería muy, muy malo. Tampoco hay un target nativo de iptables/iptables-legacy/nftables que manipule directamente el IP ID (como sí existen para el TTL, DSCP, etc). Todo esto hace que, en la práctica, sea mucho **más sencillo** (técnicamente hablando) **parchear el kernel**.

Ahora bien, en el código fuente del kernel, la parte encargada de generar el ID de los paquetes IPv4 (la cabecera IP) suele estar en el fichero:

`net/ipv4/ip_output.c`

Particularmente en la función:

`ip_select_ident_segs(…)`

…o, dependiendo de la versión:

`ip_select_ident()`

El kernel por defecto usa (simplificando) algo parecido a prandom_u32() o un algoritmo hash para inicializar y "pseudoaleatorizar" el campo id. Podríamos modificar ese código para que en lugar de generarlo aleatoriamente, simplemente lo vaya incrementando. Un ejemplo simplificado de parche (no literal, porque puede variar según la versión exacta del kernel) podría ser:

```
--- a/net/ipv4/ip_output.c
+++ b/net/ipv4/ip_output.c
@@ -XYZ,7 +XYZ,7 @@ static u16 ip_select_ident_segs(const struct sk_buff *skb, ...
{
...
- u32 ident = prandom_u32(); /* Generación aleatoria típica */
+ static u16 counter = 0;    /* O algo similar, con protección si hace wrap */
+ u32 ident = counter++;
...
return htons((u16)ident);
}
```

Se necesitarán los paquetes de desarrollo (kernel headers, toolchain, etc.) para recompilar el kernel con esos cambios. Y la tarea será bastante tediosa, porque habrá que compilar y probar cada cambio. Lógicamente esto conlleva implicaciones de seguridad considerables para toda máquina que bootee ese kernel (por ejemplo, facilitar ciertos tipos de fingerprinting y escaneos), por lo que esa máquina resultante tendría, como única función, ser un zombie de nmap.

La otra opción, y tal vez la más práctica, es instalar un Debian que useun kernel 2.2 o anterior. Por ejemplo Debian 2.2 «Potato». Debian 3.0 "Woody" (lanzada en 2002) también usaba el kernel 2.2, pero también ofrecía 2.4 como opción. Por lo que, si lo que se quiere es instalar Debian 3, habría que ver de instalar la versión que venía con el kernel 2.2.
A partir de la serie 2.4 del kernel (años 2001 a 2004), empezaron a aparecer parches que permitían elegir la aleatorización de IP ID. Eran

opcionales, pero permitían elegirlos. En la práctica, desde algún momento de la rama 2.6 (por ejemplo, alrededor de 2.6.12–2.6.20 dependiendo de la distro), ya era raro ver un comportamiento puramente incremental en un kernel "de fábrica". Hoy día (kernels 4.x, 5.x, 6.x, etc.) ya no hay opción por sysctl para volver atrás, y el código fuente implementa la randomización como comportamiento estándar.

Esta inverstigación está en desarrollo. Si quieres contribuir, ponte en contacto conmigo.