Documentación Proyecto 2- Firewall de red

1st Adrian Garcia
Departamento de Ingeniería de Sistemas
Pontificia Universidad Javeriana
Bogotá, Colombia
adriangarcia@javeriana.edu.co

2nd Nicolás Miranda
Departamento de Ingeniería de Sistemas
Pontificia Universidad Javeriana
Bogotá, Colombia
nicolasmiranda@javeriana.edu.co

3rd Miguel Romero
Departamento de Ingeniería de Sistemas
Pontificia Universidad Javeriana
Bogotá, Colombia
miguel-romero@javeriana.edu.co

I. INTRODUCCIÓN

En este proyecto se desarrolló un firewall de red y dependiendo de la dirección IP y/o protocolo, la aplicación bloqueará o permitirá la transferencia de los paquetes que sean recibidos y cumplan/incumplan con alguna expresión usada como filtro. Para esto se hizo uso de la topología propuesta en el enunciado proporcionado para la realización del proyecto.

Enseguida se describirá de manera detallada los aspectos técnicos que se tuvieron en cuenta para el desarrollo del proyecto, nombrando el lenguaje de programación y librería en el cual fue desarrollado el firewall, la lógica para la lectura de las reglas del firewall en el archivo plano, las clases y métodos que se crearon. Se referencian los RFCs de los protocolos que se usaron en el proyecto, posteriormente, una reseña breve sobre la forma en que dos firewalls existentes bloquean el tráfico de la red y el establecimiento de las reglas para estos. Finalmente, se muestran los escenarios de prueba en los que se realizaron las pruebas del firewall desarrollado.

II. DESARROLLO

Este proyecto fue desarrollado en el lenguaje Python con la versión 3.6. Para el desarrollo del firewall se utilizó la librería Pydivert [1], basada en la librería WinDivert, con la cual se pueden hacer capturas y modificaciones a los paquetes de red que se encuentra en la Windows Network Stack. Esta librería consta principalmente de 2 clases que son: WinDivert y Packet, acorde con la documentación de Pydivert. Mediante estas, se realizó todo el manejo de protocolos IP entre los que están ICMP, TCP, UDP y HTTP, teniéndose como requisito la activación de la característica IP Forwarding en el equipo donde se va a ejecutar la aplicación, ya que se usa la capa Network Forward para la captura de paquetes desde la Windows Network Stack.

El proyecto se elaboró en 3 archivos: filterReader.py, firewall.py y principal.py; donde se encuentran las clases FilterReader, Firewall y Principal, respectivamente. Estas 3 clases son las que componen el proyecto, las cuales se describen en la sección IV.

No se construyó interfaz gráfica, por lo que para la entrada de datos se usó la consola del ambiente de Python para introducir el nombre del archivo con los datos de filtrado de paquetes. Este archivo es leído con la función readRules de la clase FileReader y la expresión de filtro propuesta acorde con la estructura del archivo (Fig. 1 y Fig. 2) es posteriormente validada con la función validateExpression de la clase Firewall, acorde con las reglas de lenguaje de filtrado de WinDivert [2], librería en la cual se basa Pydivert.

```
1  #Instruction
2  block
3  #IPs
4  10.0.0.2    SrcAddr
5  20.0.0.2    SrcAddr
6  #Protocols
7  tcp DstPort==80
8  tcp SrcPort==80
9  icmp
10
```

Figura 1. Ejemplo de un archivo de entrada. Los archivos pueden estar escritos de dos maneras. Este caso corresponde a la primera manera, en la cual el archivo se distribuye de 3 secciones: la instrucción (#Instruction) que indica si se debe aceptar (allow), bloquear (block) o bloquear todo el tráfico (block all); las direcciones IP (#IPs) por las cuales se debe filtrar especificando para cada una si es dirección fuente o destino (separada la dirección de dicha especificación mediante Tab); los protocolos (#Protocols) de los paquetes que se van a filtrar que opcionalmente pueden ir acompañados de la una condición para especificar el puerto fuente o destino (separada esta condición del protocolo mediante Tab). La instrucción es obligatoria en todo caso. Si la instrucción es de bloquear todo el tráfico, las secciones posteriores no son necesarias y en caso de estar presentes no se tienen en cuenta. Si la instrucción es bloquear o aceptar se requiere al menos una de las dos secciones siguientes. La estructura general del archivo fue basada en la estructura de archivos planos manejada en archivos de secuencias biológicas donde los ítems/secciones son separados por el símbolo # y los atributos de cada registro por Tab.

```
1  block   (ip.SrcAddr == 50.0.0.2 and icmp) or (ip.DstAddr == 10.0.0.2 and udp)
2
```

Figura 2. Ejemplo de otro archivo de entrada. Los archivos pueden estar escritos de dos maneras. Este caso corresponde a la segunda manera, en la cual el archivo consiste en una única línea donde se indica primero la instrucción (allow, block o block all) y luego separado por Tab está escrita una expresión para realizar el filtrado. Esto, en el caso de requerirse controles de tráfico más complejos de lo que permitiría la escritura de un archivo de la primera manera.

Una vez validada la expresión de filtro se comparan los paquetes capturados con esta y se determina si deben ser reenviados (para el tráfico permitido) o retenidos (para el tráfico bloqueado), acorde con la instrucción especificada en el archivo de entrada. Este filtrado es realizado en tiempo real para todo el tráfico que pasa por medio de la máquina donde el firewall está instalado, manteniéndose esta tarea hasta que se decida finalizar la ejecución del programa.

Para el desarrollo de estas reglas se usó como marco conceptual base el establecimiento de las reglas de entrada y salida que utiliza Windows Defender Firewall.

III. DESCRIPCIÓN RFCs

A. RFC 791

En este documento se describe el Protocolo de Internet (IP) el cual cumple la función de transmitir datos mediante un protocolo no orientado a conexión, permite enviar información a cualquier red y su función principal es la del enrutamiento. Para este proyecto, se probaron los diferentes protocolos que están en las capas superiores a este (TCP, UDP y el ICMP) [3].

B. RFC 793

En este documento se describe el Protocolo de Control de la Transmisión (TCP) que está pensado para proporcionar un servicio de comunicación entre procesos, en un entorno con varias redes, es decir, un protocolo host a host para redes con conmutación de paquetes. Para este proyecto, se usó tanto el puerto de origen como el puerto de destino, para realizar las pruebas del firewall. Lo anterior, está propuesto en el RFC en la sección de especificación funcional, en el apartado de formato de la cabecera, donde nos indica el número de bits para el puerto y el número del mismo [4].

C. RFC 768

En este documento se describe el Protocolo de Datagrama de Usuario (UDP) el cual permite el envío de datagramas en la red sin necesidad de establecer una conexión previamente. Para este proyecto, se recalculó el checksum que es uno de los campos del encabezado de este protocolo, que sirve para verificar si el paquete llegó correctamente [5].

D. RFC 792

En este documento se describe el Protocolo de Control de Mensajes de Internet (ICMP) que en esencia sirve para el intercambio de mensajes de error, validando así, si el paquete llegó al destinatario, si el tiempo de vida (TTL), si el encabezado lleva un valor no permitido, etc. Para este proyecto, se realizó ping entre las máquinas de la red, el cual, nos mostraba toda la información en el formato de mensaje que se describe en el respectivo RFC [6].

E. RFC 2616

En este documento se describe el Protocolo de Transferencia de Hipertexto (HTTP) que funciona a nivel de aplicación el cual se puede utilizar para más funciones que solo hipertexto como por ejemplo para manejo de sistemas, nombres de servidores, códigos de error y encabezados. Para este proyecto, se configuró un servidor local web de Microsoft llamado IIS (Internet Information Services) que servía para publicar páginas web de manera local, donde se pudiese acceder a la página web de una máquina, desde otra máquina [7].

IV. RESEÑA SOBRE REGLAS DE FIREWALL

Estas reglas pueden ser definidas para el bloqueo/autorización de trafico de diferentes maneras. Una de estas es por medio del bloqueo de puertos, para que un determinado puerto no pueda hacer envío de tráfico para protocolos como TCP y UDP. Pueden también definirse reglas para el filtrado de todos los puertos de determinado protocolo, así como también de un subset de puertos para un protocolo. Otro tipo de bloqueo/autorización es por programas, el cual permite/deniega que una aplicación específica pueda enviar/recibir mensajes. En el caso del Windows Defender Firewall también se permite hacer configuraciones personalizadas adicionales como elegir si la regla aplica para red de dominio, privada o pública; y también tiene reglas predefinidas para el bloqueo de tráfico [8]. Otro ejemplo es el firewall de Norton, el cual se comporta de manera muy similar al de Windows, diferenciándose en que las reglas están diseñadas para el bloqueo de programas que pueden acceder a internet [9]. Las configuraciones realizadas en el firewall de Norton anulan las configuraciones predeterminadas que trae el mismo [10].

V. CLASES Y MÉTODOS

Para la realización de este proyecto se utilizaron tres clases: Firewall, FileReader y Principal (Fig. 3).

En la clase FileReader se creó una sola función llamada readRules, cuyo parámetro es la variable filename, que es usada para conocer la dirección y nombre del archivo (incluida la extensión) que tiene la regla que se va a definir para realizar el filtrado de tráfico. En esta función se crea una expresión de tipo String mediante concatenación por medio de la revisión del archivo de texto plano proporcionado. Una vez abierto el archivo se analiza cada una de las líneas la información existente, de acuerdo con lo especificado en Fig. 1 y Fig. 2. De forma resumida: si el archivo está vacío, se imprime un error informando esto; si se compone de una sola línea y dicha línea está escrita correctamente (Fig. 2) entonces se realiza la correspondiente lectura; si se compone de más líneas se asume una estructura acorde con Fig. 1 guardándose la instrucción especificada y concatenándose la expresión con lo que se tiene en las siguiente(s) sección(secciones). Al final, la función retorna dos parámetros: la expresión que se va a usar para filtrar el tráfico y la instrucción a usar (aceptar, bloquear o bloquear completo).

En la clase Firewall se definieron 4 métodos. El primer método (_init_) es el constructor de la clase y tiene la función de inicializar la variable expresión con la expresión que le llega por parámetro. El segundo método usa la función CheckFilter proporcionada por la librería para chequear si la expresión de filtrado expresada en el archivo de texto es correcta de acuerdo

con las reglas de filtrado de WinDivert, como se mencionó anteriormente, utilizando el tercer retorno de la función CheckFilter, por tanto, retornando el String “No error” si la expresión es correcta o “Error” si presenta errores y por tanto no es posible usarla. La tercera función es allowTraffic la cual filtra todo el tráfico de la capa Network Forward y luego para cada paquete usando la función matches de la librería determina si el paquete coincide con la expresión, en caso de que sí coincida el paquete se reenvía con su checksum recalculado usando la función send de la librería. La función blockTraffic al igual que la anterior, también filtra todo el tráfico y para cada paquete compara la expresión, en caso de no coincidir con esta el paquete es reenviado. Las funciones allowTraffic y blockTraffic filtran el tráfico en tiempo real, ejecutándose indefinidamente hasta que el usuario decida finalizar la ejecución.

La tercera y última clase, es la clase Principal, la cual cuenta con un método llamado main desde el cual se llevan se solicita al usuario proporcionar la ruta relativa del archivo de texto con la regla para filtrar el tráfico, posteriormente se valida si el archivo existe, luego se usa una instancia de FileReader para leer el archivo y si la expresión está presente. Luego, utilizando una instancia de Firewall se revisa si la expresión es correcta. Por último, en caso de pasar todas las validaciones, se realizan llamados se determina cuál es la instrucción y posteriormente se hace llamado a la función de firewall respectiva, es decir, allowTraffic o blockTraffic.

VI. DIAGRAMAS UML

Se realizó tres diagramas UML (Lenguaje Unificado de Modelado), que sirve para mostrar la arquitectura, diseño e implementación de un sistema de software de manera visual [11].

A. Diagrama de clases

Muestra las clases que hay dentro del programa, escribiendo tanto los atributos, métodos y las relaciones que hay entre

clases. En términos generales, describe de manera estática la estructura del programa.

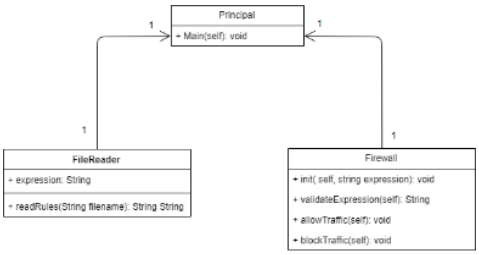


Figura 3. Diagrama de clases del firewall desarrollado.

B. Diagrama de actividad

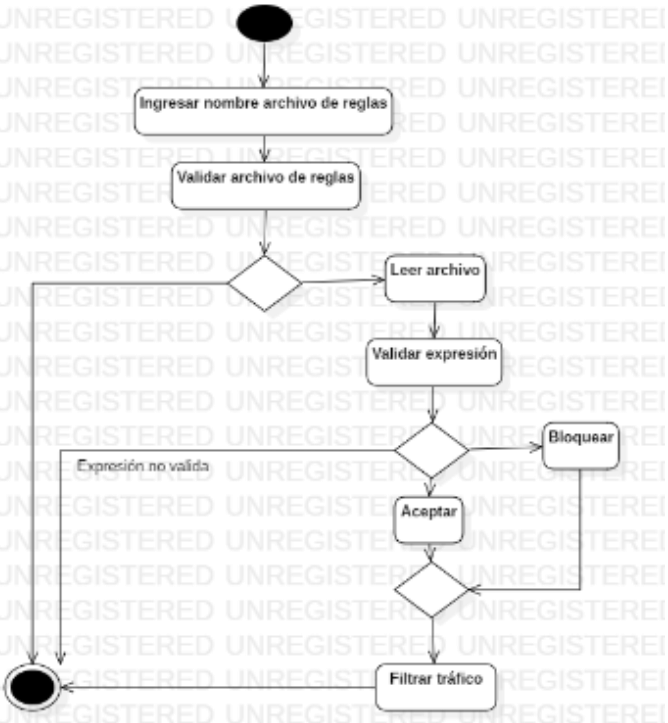
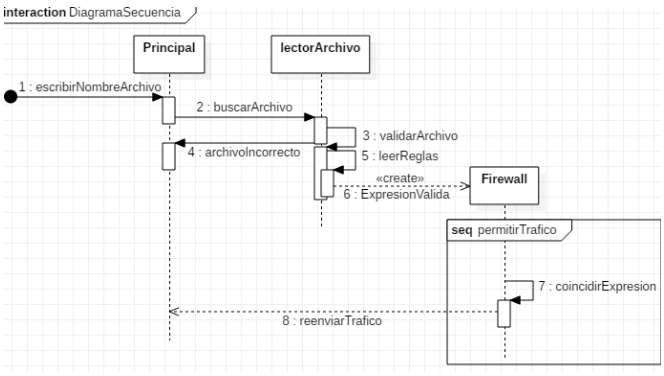


Figura 4. Diagrama de actividades del firewall desarrollado.

C. Diagrama de secuencia

Muestra la interacción entre los objetos y el orden de cómo ocurren los eventos con base en el tiempo de vida de cada uno.



VII. ESCENARIO DE PRUEBA

Para la realización de pruebas para este proyecto se enviaron diferentes tipos de mensajes en los cuales se cambiaba el tipo del protocolo. Para las pruebas del protocolo ICMP se realizaron pings. Para los protocolos TCP y UDP se utilizó la aplicación PacketSender la cual permite el envío de ese tipo de paquetes conociendo la dirección IP destino y eligiendo un puerto de destino.

A continuación, se va a presenta de manera detallada tres escenarios de prueba donde se evidencia el correcto funcionamiento del firewall, realizando pruebas tanto de bloque de direcciones IP, como de protocolos y como de puertos.

A. Bloqueo por protocolo ICMP

En la Fig. 6 se muestra la consola de comandos que ejecuta de manera correcta el ping desde la máquina 10.0.0.2 para probar el protocolo ICMP antes de correr el firewall.

```
C:\Users\redes>ping 50.0.0.2

Haciendo ping a 50.0.0.2 con 32 bytes de datos:
Respuesta desde 50.0.0.2: bytes=32 tiempo=1ms TTL=126
Respuesta desde 50.0.0.2: bytes=32 tiempo=1ms TTL=126
Respuesta desde 50.0.0.2: bytes=32 tiempo=1ms TTL=126
Respuesta desde 50.0.0.2: bytes=32 tiempo=1ms TTL=126

Estadísticas de ping para 50.0.0.2:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 1ms, Máximo = 1ms, Media = 1ms

C:\Users\redes>ping 20.0.0.2

Haciendo ping a 20.0.0.2 con 32 bytes de datos:
Respuesta desde 20.0.0.2: bytes=32 tiempo=1ms TTL=127
Respuesta desde 20.0.0.2: bytes=32 tiempo<1m TTL=127
Respuesta desde 20.0.0.2: bytes=32 tiempo<1m TTL=127
Respuesta desde 20.0.0.2: bytes=32 tiempo<1m TTL=127

Estadísticas de ping para 20.0.0.2:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 0ms, Máximo = 1ms, Media = 0ms

C:\Users\redes>ping 30.0.0.1

Haciendo ping a 30.0.0.1 con 32 bytes de datos:
Respuesta desde 30.0.0.1: bytes=32 tiempo<1m TTL=128
Respuesta desde 30.0.0.1: bytes=32 tiempo<1m TTL=128
Respuesta desde 30.0.0.1: bytes=32 tiempo<1m TTL=128
Respuesta desde 30.0.0.1: bytes=32 tiempo<1m TTL=128

Estadísticas de ping para 30.0.0.1:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 0ms, Máximo = 0ms, Media = 0ms
```

Figura 6. ICMP antes de ejecutar el firewall.

Posteriormente, en la Fig. 7 se muestra el formato de las reglas establecidas y en la Fig. 8, se muestra el programa en el momento que lee el archivo de texto con las reglas.

```
#Instruction
block
#Protocols
icmp
```

Figura 7. Instrucciones para bloquear protocolo ICMP.

```
Write the file name with extension (e.g. rules.txt):

blockICMP.txt

File with rules read

Expression to filter: icmp
```

Figura 8. Lectura del archivo en el programa.

Tan pronto cuando se ejecuta el programa, se evidencia que en Wireshark de la máquina 10.0.0.2 no captura ningún paquete (Fig. 9) y en la máquina 30.0.0.1 que es la que tiene

configuradas las tres IPs, intercepta los mensajes, pero no los reenvía (Fig, 10).

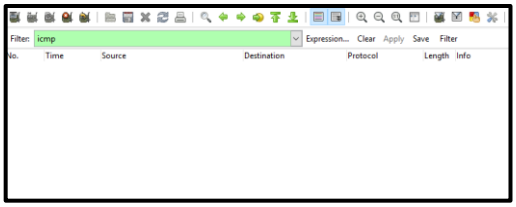


Figura 9. Captura Wireshark, host 10.0.0.2.

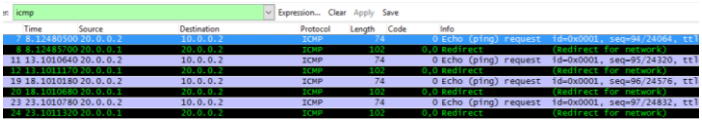


Figura 10. Captura Wireshark, host 30.0.0.1.

B. Bloqueo por protocolo HTML

En la Fig.11 se muestra la página web de las redes internas que tienen IP 10.0.0.2 y 20.0.0.2 respectivamente, funcionando de manera correcta, sin embargo, cuando activamos el firewall para filtrar todo el tráfico donde la destino sea IP 50.0.0.2 y el puerto sea el 80 del protocolo TCP (Fig. 12) se cae la página (Fig. 13) y se puede observar mediante la captura de paquetes de Wireshark que se está intentando hacer la conexión pero no deja realizar el Three Way Handshake completo, solo intenta [SYN] sin recibir alguna respuesta (Fig. 14).

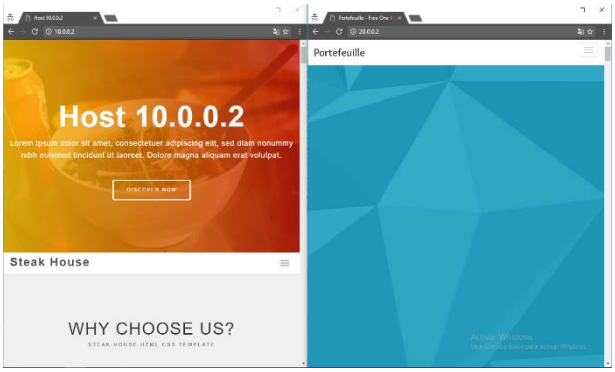


Figura 11. Página web funcionando correctamente.

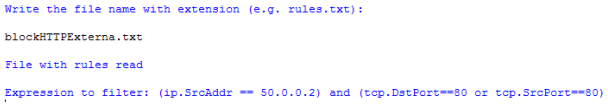


Figura 12. Lectura del archivo en el programa.

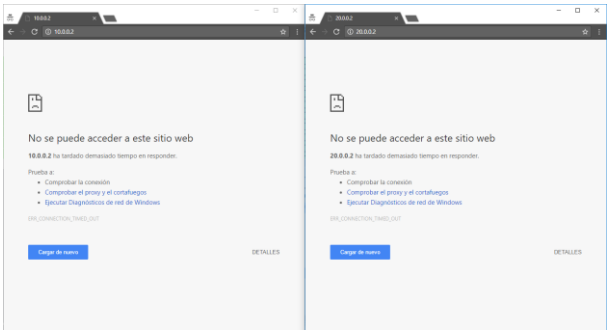


Figura 13. Página web caída.

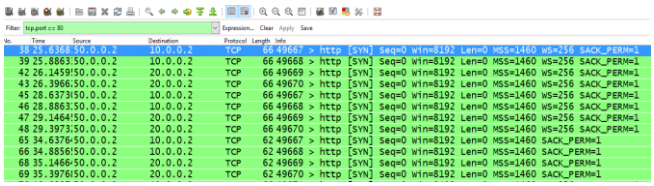


Figura 14. Captura del tráfico en Wireshark.

C. Bloqueo por IP

Para este caso de prueba se decidió bloquear todo el tráfico que fuese y viniese desde la IP 10.0.0.2. En la Fig. 15 se puede ver que Wireshark está capturando todos los paquetes de los diferentes protocolos que vienen y salen de esta IP, después, cuando activamos el firewall (Fig. 16), se puede observar en la Fig. 17 que la IP 10.0.0.2 ya no recibe ni envía ningún paquete hacia las otras redes.

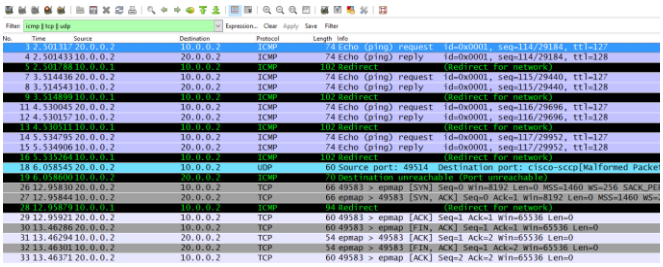


Figura 15. Wireshark capturando paquetes antes de inicializar el programa.

```
Write the file name with extension (e.g. rules.txt):

blockIp10.txt

File with rules read

Expression to filter: (ip.SrcAddr == 10.0.0.2 or ip.DstAddr == 10.0.0.2
```

Figura 16. Lectura del archivo en el programa.

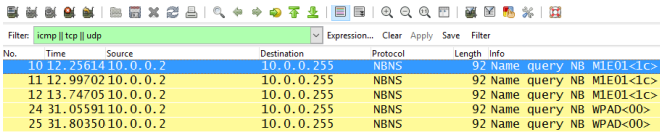


Figura 15. Wireshark capturando paquetes después de inicializar el programa.

VIII. CONCLUSIONES

Los firewalls son una herramienta bastante útil en el ámbito de las redes ya que permiten tener el control de estas mediante las reglas que sean configuradas. Es una ayuda para prevenir ataques de redes externas si se configura de manera correcta. Además, permiten bloquear la comunicación con sitios que no son deseables o bloquear la comunicación con otras partes de la red si se considera necesario. Por medio del bloqueo de ciertos protocolos y/o puertos se pueden prevenir ataques a la red.

En relación con los temas visto en clase, se pudo poner en prácticas los bastante de ellos, tales como: capas del modelo OSI y modelo TCP/IP, protocolos de cada unas de las capas del modelo propuesto en el libro Redes de computadores [12] y cada uno de los campos y especificaciones funcionales que tienen.

IX. REFERENCIAS

[1] F. Falcinelli, “Pydivert API Reference,” 2018. [Online]. Available: https://github.com/ffalcinelli/pydivert. [Accessed: 22-Nov-2018].

[2] WinDivert, “WinDivert 1.4: Windows Packet Divert,” 2018. [Online]. Available: https://ffalcinelli.github.io/pydivert/. [Accessed: 22-Nov-2018].

[3] U. of S. California, “Internet Protocol,” 1981. [Online]. Available: https://tools.ietf.org/html/rfc791. [Accessed: 22-Nov-2018].

[4] U. of S. California, “Transmission Control Protocol,” 1981. [Online]. Available: https://tools.ietf.org/html/rfc793. [Accessed: 22-Nov-2018].

[5] J. Postel, “User Datagram Protocol,” 1980. [Online]. Available: https://tools.ietf.org/html/rfc768.

[6] J. Postel, “Internet Control Message Protocol,” 1981. [Online]. Available: https://tools.ietf.org/html/rfc792. [Accessed: 22-Nov-2018].

[7] R. Fielding, “Hypertext Transfer Protocol -- HTTP/1.1,” 1999. [Online]. Available: https://tools.ietf.org/html/rfc2616. [Accessed: 22-Nov-2018].

[8] Microsoft, “Configuracón avanzada Firewall windows,” 2018. [Online]. Available: https://msdn.microsoft.com/es-es/library/windows/hardware/dn898511(v=vs.85).aspx.

[9] Norton, “Firewall inteligente.” 2018.

[10] Norton, “Cómo agregar un programa a Control de programas,” 2018. [Online]. Available: https://support.norton.com/sp/es/es/home/current/solutions/v1028102_NIS_Retail_2015_es_es?

[11] Lucidchart, “Qué es el lenguaje unificado de modelado (UML),” 2018. [Online]. Available: https://www.lucidchart.com/pages/es/que-es-el-lenguaje-unificado-de-modelado-uml. [Accessed: 22-Nov-2018].

[12] A. S. Tanenbaum and D. J. Wetherall, *Redes De Computadoras*. 2012.