Reingeniería de Políticas de Firewall

Luciano Peschiutta[1][0009−0000−1574−4297], Bruno A. Genero[1][0009−0005−1161−5101], Miguel Solinas[1][0000−0002−7550−1067], and Marcos Bettucci[2][0009−0005−1778−1952]

[1]LARYC, Facultad de Ciencias Exactas Físicas y Naturales, Universidad Nacional de Córdoba, AR {luciano.peschiutta, bruno.genero, miguel.solinas}@unc.edu.ar
[2]Prosecretaría de Informática, Universidad Nacional de Córdoba, AR
{mbettucci}@unc.edu.ar

Resumen Los firewalls desempeñan un rol crítico en la seguridad de redes, actuando como la primera línea de defensa, filtrando el tráfico entrante y saliente según un conjunto de reglas predefinidas. La adición secuencial de reglas con el tiempo genera inconsistencias, redundancias y degradación del rendimiento. Este trabajo presenta una metodología para la reingeniería de políticas de firewall, buscando alcanzar tres propiedades fundamentales: consistencia, completitud y compacidad. Mediante la utilización de los Firewall Decision Diagrams (FDD) se proporciona un enfoque estructurado para optimizar conjuntos de reglas, minimizando conflictos y mejorando la mantenibilidad.

Keywords: Ciberseguridad · Seguridad en Redes · Firewalls

Firewall Policy Reengineering

Abstract Firewalls play a critical role in network security, acting as the first line of defense, filtering incoming and outgoing traffic based on a set of predefined rules. The sequential addition of rules over time leads to inconsistencies, redundancies, and performance degradation. This paper presents a methodology for firewall policy reengineering, seeking to achieve three fundamental properties: consistency, completeness, and compactness. By utilizing Firewall Decision Diagrams (FDDs), a structured approach is provided for optimizing rule sets, minimizing conflicts and improving maintainability.

Keywords: Cibersecurity · Network Security · Firewalls.

1 Introducción

Los firewalls son componentes esenciales en la seguridad de redes, encargados de filtrar el tráfico basado en un conjunto de reglas predefinidas y expresión de una política de firewall. Su función principal es examinar los paquetes que transitan

por la red de una organización y decidir si deben ser aceptados o descartados. En organizaciones con redes complejas, los firewalls acumulan un gran número de reglas. Esto dificulta su comprensión y mantenimiento, lo que demanda una reestructuración significativa y periódica de las mismas [2]. Esta reingeniería de reglas de un firewall presenta riesgos de generar incoherencias, conflictos y posibles brechas de seguridad, comprometiendo su eficacia. El nivel de seguridad de un firewall no está garantizado con su costo, sino que proviene esencialmente de la gestión de la configuración de reglas.

Para este trabajo consideramos a un paquete como una tupla con un número finito de campos. La función de un firewall se especifica formalmente como una secuencia de reglas, en la que cada una de ellas tiene la forma:

$$\texttt{<predicado>} \rightarrow \texttt{<decisión>}$$

El `<predicado>` de una regla es una expresión booleana que aplica sobre un conjunto de campos de un paquete de red. La `<decisión>` de una regla es la acción a realizar sobre el paquete de red que coincide con la expresión del predicado. Un paquete coincide con una regla si y sólo si satisface el predicado de la regla.

Las reglas en un firewall a menudo entran en conflicto. Se dice que dos reglas en un firewall entran en conflicto si se superponen. Dos reglas en un firewall se superponen si hay al menos un paquete que pueda coincidir con ambas reglas. Para resolver este tipo de conflicto, la decisión para cada paquete es la decisión de la primera regla (la regla de mayor prioridad) que coincida con el paquete. Esto hace que el orden de las reglas sea un aspecto crítico en la configuración y mantenimiento del firewall.

Netfilter es el framework del kernel de Linux encargado del filtrado y manipulación de paquetes de red. Actúa como el núcleo del sistema de firewall en Linux y ofrece puntos de control en diferentes etapas del tráfico [5]. Herramientas como iptables y nftables funcionan sobre la base de Netfilter y permiten definir reglas de filtrado, NAT y seguimiento de conexiones. Nftables, introducido como sucesor de iptables, ofrece una sintaxis más eficiente y un control unificado sobre las reglas de filtrado.

Dado que la motivación de este trabajo se inició en un requerimiento de reingeniería de políticas de un firewall basado en host, se utilizará Netfilter como base para validar la propuesta. El objetivo de esta reingeniería de políticas es mejorar su completitud, compacidad y consistencia mediante técnicas de optimización y reestructuración. El procedimiento desarrollado no se limita exclusivamente a Netfilter, sino que es extensible a otros sistemas basados en reglas, como firewalls que utilicen listas de control de acceso (ACLs) o sus implementaciones en routers y switches.

Debido a los conflictos y sensibilidad al orden que presentan las reglas que conforman una política de filtrado [4], diseñar un firewall directamente como una secuencia de reglas presenta tres problemas: el problema de consistencia, el problema de completitud y el problema de compacidad [3].

- La **consistencia** de un conjunto de reglas de firewall se define como la ausencia de conflictos entre sus reglas. Esto significa que no puede haber dos reglas que se apliquen al mismo paquete y que lleven a decisiones contradictorias.
- La propiedad de **completitud** indica que el conjunto de reglas puede clasificar toda combinación posible de valores que los campos de un paquete de red puedan tener. Esto implica que para cada posible paquete que llegue al firewall, existiría al menos una regla que lo maneje.
- La **compacidad** de un conjunto de reglas se cumple si cada regla del conjunto es esencial y no hay reglas que se puedan eliminar sin cambiar el comportamiento del firewall. Un firewall es compacto si no contiene reglas redundantes o innecesarias.

Este trabajo se ha organizado de la siguiente manera. La Sección 2 presenta una revisión de la literatura existente sobre **Firewall Decision Diagrams**. La Sección 3 describe el desarrollo de una aplicación que permite la reingeniería de políticas de firewall. La Sección 4 presenta y analiza los resultados obtenidos. La Sección 5 ofrece una discusión de las implicaciones de estos resultados y en la Sección 6 se encuentran las conclusiones del trabajo.

2 Firewall Decision Diagrams

Para abordar los desafíos de configuración y mantenimiento de reglas de un firewall y buscar el cumplimiento de las propiedades de consistencia, completitud y compacidad, en este trabajo se propone el uso de Firewall Decision Diagrams (FDD) [3] como una representación estructurada de las reglas del firewall. A diferencia de la representación tradicional basada en secuencias de reglas, los FDD permiten modelar las decisiones de filtrado como estructuras gráficas optimizadas. La propia semántica del FDD elimina la posibilidad de que existan redundancias y conflictos, también facilita la identificación de los comportamientos de la política de firewall. Mediante este enfoque, se logra una administración más efectiva de políticas de firewall, asegurando su corrección y eficacia.

Un *Campo* F_i es una variable cuyo dominio, denotado $D(F_i)$, es un intervalo finito de enteros no-negativos. Un *Paquete* sobre los campos $F_1, \ldots, F_d$ es una d-tupla $(p_1, \ldots, p_d)$ donde cada $p_i (1 <= i <= d)$ es un elemento de $D(F_i)$. Usamos Σ para denotar al set de todos los paquetes sobre los campos $F_1, \ldots, F_d$.

Un *Firewall Decision Diagram* (FDD) f sobre los campos $F_1, \ldots, F_d$ es un *Grafo Acíclico Dirigido* (DAG) que tiene las siguientes propiedades:

1. Hay exactamente un nodo en f que no tiene arcos entrantes. Este nodo se denomina la *raíz* de f. Los nodos en f que no tienen arcos salientes se llaman nodos *terminales* de f.
2. Cada nodo v en f esta etiquetado con un campo, denotado $F(v)$, tal que:

$$F(v) \in \begin{cases} \{F_1, \ldots, F_d\} & \text{si } v \text{ es no-terminal.} \\ \{Aceptar, Descartar\} & \text{si } v \text{ es terminal.} \end{cases}$$

3. Cada arco e en f está etiquetado con un set no-vacío de enteros, denotado $I(e)$, tal que si e es un arco de salida del nodo v, entonces tenemos $I(e) \subseteq D(F(v))$.
4. Un camino dirigido en f desde el nodo raíz a un nodo terminal se llama *camino de decisión*. Ningún par de nodos en un camino de decisión tienen las mismas etiquetas.
5. El set de todos los arcos de salida de un nodo v en f, denotado $E(v)$, satisface las siguientes dos condiciones:
   (a) *Consistencia*: $I(e) \cap I(e') = \emptyset$ para cualquier par de arcos distintos e y e' en $E(v)$.
   (b) *Completitud*: $\cup_{e \in E(v)} I(e) = D(F(v))$.

La Fig. 1 muestra un ejemplo de FDD sobre 2 campos F1 y F2. El dominio de cada campo es el intervalo [1, 10]. Nótese que en el etiquetado de los nodos terminales se usan las letras “a” y “d” para representar las decisiones de “Aceptar” y “Descartar”, respectivamente.

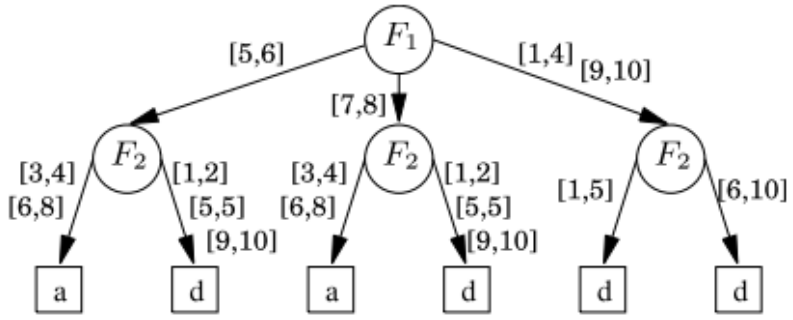

Fig. 1. Ejemplo de FDD. Fuente: [3]

Un FDD asigna a cada paquete una decisión mediante la prueba del mismo a lo largo del diagrama, desde la raíz hasta un nodo terminal, que indica la decisión del firewall para el paquete. Cada nodo no terminal en un FDD especifica una prueba de un campo del paquete, y cada arco saliente de ese nodo corresponde a un conjunto de valores posibles de ese campo. Cada paquete se asigna a una decisión comenzando en la raíz, probando el campo que etiqueta este nodo y luego moviéndose hacia abajo por la arista cuya etiqueta contiene el valor del campo del paquete. Este proceso se repite para el sub-diagrama que tiene como raíz el nuevo nodo.

Para cada paquete p, hay una y solo una regla que se empareja con p debido a las propiedades de consistencia y completitud de un FDD. Por ejemplo, las reglas representadas por todas los caminos de decisiones en el FDD de la Fig. 1

se listan el la Fig. 2 Tomando como ejemplo el paquete (7,9), solo se empareja con la regla r4 en la Fig. 2.

$$
\begin{aligned}
r_1\colon\ & F_1 \in [5,6] \wedge F_2 \in [3,4] \cup [6,8] && \rightarrow a\\
r_2\colon\ & F_1 \in [5,6] \wedge F_2 \in [1,2] \cup [5,5] \cup [9,10] && \rightarrow d\\
r_3\colon\ & F_1 \in [7,8] \wedge F_2 \in [3,4] \cup [6,8] && \rightarrow a\\
r_4\colon\ & F_1 \in [7,8] \wedge F_2 \in [1,2] \cup [5,5] \cup [9,10] && \rightarrow d\\
r_5\colon\ & F_1 \in [1,4] \cup [9,10] \wedge F_2 \in [1,5] && \rightarrow d\\
r_6\colon\ & F_1 \in [1,4] \cup [9,10] \wedge F_2 \in [6,10] && \rightarrow d
\end{aligned}
$$

Fig. 2. Todas las reglas representadas por el FDD en la Fig. 1. Fuente: [3].

3 Contribución

En el contexto de un Proyecto Integrador (tesina de grado) de Ingeniería en Computación de la Universidad Nacional de Córdoba, se desarrolló una aplicación que permite trabajar con políticas de firewall implementadas mediante sistemas ordenados de reglas (inicialmente bajo el modelo de iptables). Esta herramienta ofrece las siguientes funcionalidades principales:

- Reestructurar la política para garantizar que cumpla con las propiedades de completitud , consistencia y compacidad.
- Analizar y representar explícitamente el comportamiento actual del firewall, facilitando su comprensión por parte de administradores y operadores.
- Modificar la política de forma asistida, permitiendo la inserción, eliminación o modificación de reglas sin requerir un análisis exhaustivo del comportamiento previo ni un conocimiento detallado del ordenamiento de reglas necesario para mantener la coherencia de la política.

Para optimizar automáticamente la configuración del firewall respetando las propiedades mencionadas, la aplicación implementa el siguiente flujo de transformación:

- Convertir el conjunto original de reglas (con semántica de iptables) a un Firewall Decision Diagram (FDD).
- Aplicar técnicas de optimización sobre el FDD, eliminando reglas redundantes, resolviendo conflictos y compactando su estructura sin alterar su comportamiento funcional.
- Generar un nuevo conjunto de reglas a partir del FDD optimizado, manteniendo la semántica de iptables para que pueda ser aplicado directamente en el sistema de destino.

Dadas las ventajas del uso de FDDs para representar, visualizar y gestionar políticas de firewall, la aplicación también permite que, una vez realizada la transformación, el usuario continúe administrando la política directamente sobre el modelo FDD. En este sentido, se ofrece funcionalidad para:

- Visualizar gráficamente la estructura del FDD, incluyendo los nodos de decisión y los caminos asociados a acciones específicas (permitir, denegar, etc), con el fin de entender cómo se procesa cada paquete.
- Modificar el FDD, permitiendo agregar nuevas condiciones de decisión, eliminar nodos o cambiar las acciones asociadas, lo que equivale a insertar, eliminar o modificar reglas en el conjunto de políticas del firewall.

Para el caso de prueba de filtrado se utilizó la sintaxis básica de iptables utilizando los siguientes atributos básicos : ip de origen, ip de destino, puerto de origen, puerto de destino y protocolo.

El sistema se encarga de tomar el contenido de un archivo de texto, ver la Fig. 3, con la estructura de configuración de un firewall. Luego realiza la transformación para obtener su representación como FDD, optimizarla y visualizarla (Fig. 4). Finalmente, se reconvierte a la sintaxis original y se exporta a otro archivo con un formato de firewall específico.

```
-A INPUT   -s 1.1.1.0/24    -d 2.2.2.0/24    -p udp -j DROP
-A INPUT   -s 1.1.1.128/25  -d 3.3.3.0/24    -p tcp -j ACCEPT
-A INPUT   -s 1.1.1.128/25  -d 3.3.3.0/25    -p udp -j DROP
-A INPUT   -s 1.1.1.128/25  -d 3.3.3.64/30   -p udp -j ACCEPT
-A INPUT   -s 1.1.1.128/25  -d 3.3.3.68/30   -p udp -j DROP
-A INPUT   -s 5.5.5.0/25    -d 3.3.3.68/30   -p udp -j DROP
```

Fig. 3. Cadena de ejemplo de iptables

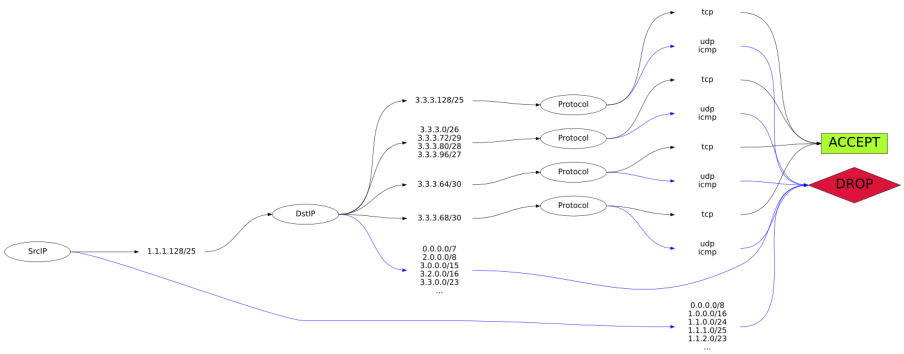

Fig. 4. FDD equivalente

La aplicación tiene un diseño modular, con la flexibilidad necesaria para adaptarse a diferentes sintaxis de firewalls que gestionan políticas como listas de reglas. Los módulos representados en la Fig. 5 desempeñan las siguientes funciones:

1. **Rules:** Se encarga de representar tanto las reglas ingresadas al sistema como las versiones optimizadas generadas. Es completamente independiente de las diferentes sintaxis de entrada y salida, lo que lo hace adaptable a distintos entornos.
2. **Parser:** Su función es adaptar las reglas de entrada al formato interno requerido. Implementa estrategias para transformar reglas de diversas sintaxis al formato requerido por el módulo Rules y, de forma inversa, devuelve las reglas optimizadas al formato original.
3. **Fields:** Define los campos que el sistema está preparado para gestionar, junto con los conjuntos de elementos asociados y las operaciones posibles entre ellos. Este módulo permite la expansión del sistema mediante la incorporación de nuevas subclases de conjuntos de elementos y la configuración de campos adicionales, habilitando así nuevas funcionalidades.
4. **FDD:** Proporciona las herramientas necesarias para generar, visualizar y optimizar Firewall Decision Diagrams, así como para aplicar filtros que mejoren su visualización.
5. **Firewall:** Este módulo actúa como el núcleo del sistema, integrando todos los módulos anteriores y proporcionando las interfaces necesarias para su comunicación.
6. **Model, View y Controller (MVC):** Representan la implementación del clásico patrón de diseño MVC, utilizado para gestionar la aplicación de manera estructurada.

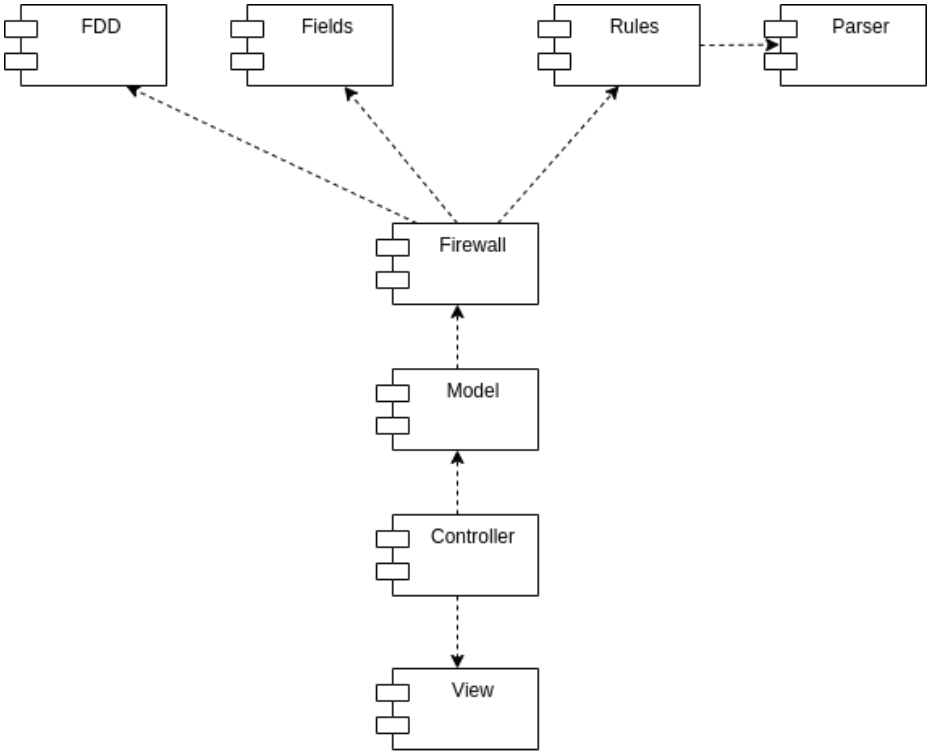

Fig. 5. Arquitectura de la aplicación

La visualización de FDD resulta poco práctica en políticas de firewall que superan la cantidad de 50 reglas dado que la cantidad de elementos hace casi imposible seguir los caminos de decisión. Para resolver este problema, se añadió la capacidad de filtrar elementos que permitan enfocarse en las partes del diagrama que se desean analizar en un momento determinado. En la Fig. 6 podemos observar una versión filtrada de la Fig. 2 para indicar solo aquellos caminos de decisión que contengan la dirección de destino 3.3.3.130.

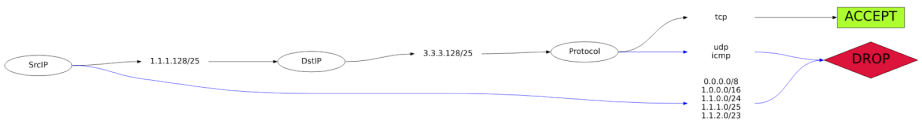

Fig. 6. FDD filtrado para dirección de destino 3.3.3.130

La aplicación se diseñó como una Integrated Development Environment (IDE) para que los usuarios puedan manipular y visualizar las políticas de un firewall.

Está compuesta por varios elementos que permiten la interacción con el sistema, como su edición, optimización y reordenamiento de forma visual. En la Fig. 7 se muestra una captura de pantalla.

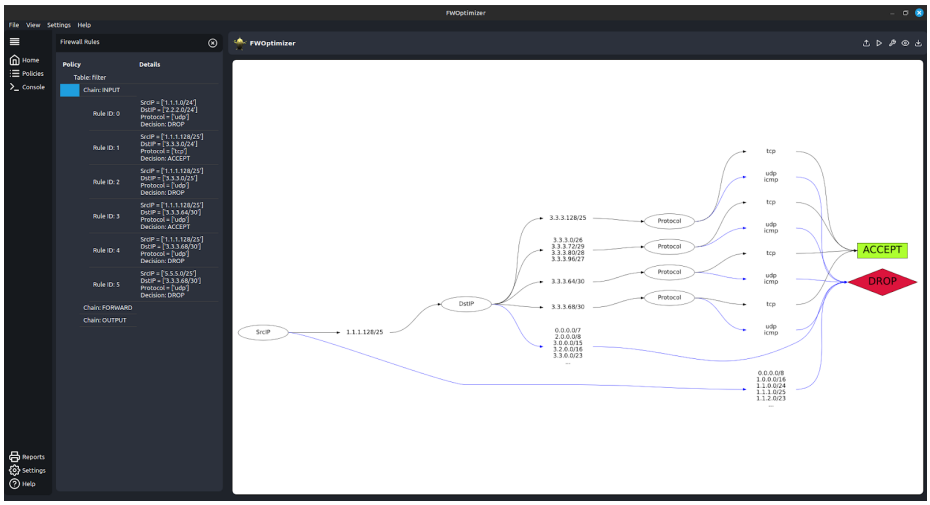

Fig. 7. Aplicación en funcionamiento

3.1 Resultados

Para evaluar la efectividad de la herramienta en los procesos de optimización, se utilizaron varias políticas de firewall hasta un conjunto máximo de 87 reglas obtenidos del estudio de casos presentados en [1] (Ver Table 1 y Table 2). Este conjunto de reglas representa una política de firewall real que contiene redundancias, inconsistencias y posibles conflictos, lo que lo convierte en un caso adecuado para una primera validación del enfoque propuesto.

#	**IPTables Rule**
1	-A INPUT -s 67.54.138.163 -d 157.96.119.153 -p tcp -m tcp –dport 9100 -j ACCEPT
2	-A INPUT -s 67.54.138.163 -d 157.96.119.153 -p udp -m udp –dport 161 -j ACCEPT
3	-A INPUT -p udp -m udp –dport 53 -j DROP
4	-A INPUT -p udp -m udp –dport 55 -j DROP
5	-A INPUT -p udp -m udp –dport 77 -j DROP
6	-A INPUT -d 157.96.252.66 -j ACCEPT
7	-A INPUT -s 32.45.186.83 -j DROP
8	-A INPUT -d 157.96.139.14/32 -p tcp -m tcp –dport 443 -j DROP
9	-A INPUT -s 231.49.182.251 -j DROP
10	-A INPUT -p tcp -m tcp –dport 3127 -j DROP
11	-A INPUT -p tcp -m tcp –dport 2745 -j DROP
12	-A INPUT -p udp -m udp –dport 4000 -j DROP
13	-A INPUT -p udp -m udp –dport 111 -j DROP
14	-A INPUT -p tcp -m tcp –dport 111 -j DROP
15	-A INPUT -p udp -m udp –dport 2049 -j DROP
16	-A INPUT -p tcp -m tcp –dport 2049 -j DROP
17	-A INPUT -p udp -m udp –dport 7 -j DROP
18	-A INPUT -p tcp -m tcp –dport 7 -j DROP
19	-A INPUT -p tcp -m tcp –dport 6346 -j DROP
20	-A INPUT -p tcp -m tcp –dport 7000 -j DROP
21	-A INPUT -p udp -m udp –dport 161 -j DROP
22	-A INPUT -p udp -m udp –dport 162 -j DROP
23	-A INPUT -p udp -m udp –dport 1993 -j DROP
24	-A INPUT -p udp -m udp –dport 67 -j DROP
25	-A INPUT -p udp -m udp –dport 68 -j DROP
26	-A INPUT -p udp -m udp –dport 49 -j DROP
27	-A INPUT -s 178.95.49.0/24 -j DROP
28	-A INPUT -s 157.96.119.0/24 -j DROP
29	-A INPUT -s 157.96.120.0/24 -j DROP
30	-A INPUT -s 157.96.121.0/24 -j DROP
31	-A INPUT -s 157.96.122.0/24 -j DROP
32	-A INPUT -s 157.96.130.0/24 -j DROP
33	-A INPUT -s 157.96.138.0/24 -j DROP
34	-A INPUT -s 157.96.139.0/24 -j DROP
35	-A INPUT -s 157.96.143.0/24 -j DROP

Table 1. IPTables Firewall Rules (Part 1)

#	IPTables Rule
36	-A INPUT -s 157.96.144.0/24 -j DROP
37	-A INPUT -s 157.96.158.0/24 -j DROP
38	-A INPUT -s 157.96.252.0/24 -j DROP
39	-A INPUT -d 157.96.139.9 -p udp -m udp –dport 1949 -j ACCEPT
40	-A INPUT -d 157.96.139.10 -p udp -m udp –dport 1949 -j ACCEPT
41	-A INPUT -d 157.96.120.2 -p udp -m udp –dport 1949 -j ACCEPT
42	-A INPUT -d 157.96.139.9 -p tcp -m tcp –dport 1949 -j ACCEPT
43	-A INPUT -d 157.96.139.10 -p tcp -m tcp –dport 1949 -j ACCEPT
44	-A INPUT -d 157.96.120.2 -p tcp -m tcp –dport 1949 -j ACCEPT
45	-A INPUT -s 255.255.255.255/32 -j DROP
46	-A INPUT -s 0.0.0.0/32 -j DROP
47	-A INPUT -s 157.96.119.0/24 -j DROP
48	-A INPUT -d 157.96.139.10 -p tcp -m tcp –dport 109 -j ACCEPT
49	-A INPUT -d 157.96.139.10 -p tcp -m tcp –dport 110 -j ACCEPT
50	-A INPUT -d 157.96.139.10 -p tcp -m tcp –dport 143 -j ACCEPT
51	-A INPUT -s 62.78.103.0/24 -j DROP
52	-A INPUT -p tcp -m tcp –dport 6667 -j DROP
53	-A INPUT -p tcp -m tcp –dport 6112 -j DROP
54	-A INPUT -p tcp -m tcp –dport 109 -j DROP
55	-A INPUT -p tcp -m tcp –dport 110 -j DROP
56	-A INPUT -p udp -m udp –dport 1433 -j DROP
57	-A INPUT -p udp -m udp –dport 1434 -j DROP
58	-A INPUT -p tcp -m tcp –dport 135 -j DROP
59	-A INPUT -p tcp -m tcp –dport 137 -j DROP
60	-A INPUT -p tcp -m tcp –dport 138 -j DROP
61	-A INPUT -p tcp -m tcp –dport 139 -j DROP
62	-A INPUT -p tcp -m tcp –dport 445 -j DROP
63	-A INPUT -p udp -m udp –dport 135 -j DROP
64	-A INPUT -p udp -m udp –dport 137 -j DROP
65	-A INPUT -p udp -m udp –dport 138 -j DROP
66	-A INPUT -p udp -m udp –dport 139 -j DROP
67	-A INPUT -p udp -m udp –dport 445 -j DROP
68	-A INPUT -p tcp -m tcp –dport 143 -j DROP
69	-A INPUT -p tcp -m tcp –dport 515 -j DROP
70	-A INPUT -p tcp -m tcp –dport 512 -j DROP
71	-A INPUT -p udp -m udp –dport 514 -j DROP
72	-A INPUT -p udp -m udp –dport 69 -j DROP
73	-A INPUT -p tcp -m tcp –dport 514 -j DROP
74	-A INPUT -s 157.96.138.138/32 -p tcp -m tcp –dport 5900 -j ACCEPT
75	-A INPUT -s 157.96.138.138/32 -p tcp -m tcp –dport 5166 -j ACCEPT
76	-A INPUT -s 157.96.138.138/32 -j DROP
77	-A INPUT -s 157.96.138.101/32 -p tcp -m tcp –dport 5900 -j ACCEPT
78	-A INPUT -s 157.96.138.101/32 -p tcp -m tcp –dport 5166 -j ACCEPT
79	-A INPUT -s 157.96.138.101/32 -j DROP
80	-A INPUT -s 157.96.138.80/32 -j DROP
81	-A INPUT -s 157.96.138.82/32 -j DROP
82	-A INPUT -s 157.96.138.234/32 -j DROP
83	-A INPUT -s 157.96.138.235/32 -j DROP
84	-A INPUT -s 157.96.138.236/32 -j DROP
85	-A INPUT -s 157.96.128.0/24 -j ACCEPT
86	-A INPUT -s 157.96.140.0/24 -j DROP
87	-A INPUT -j ACCEPT

Table 2. IPTables Firewall Rules (Part 2)

Tras procesar este conjunto de reglas con nuestra herramienta, se identificaron y corrigieron reglas redundantes e inconsistentes. En particular las que se muestran a continuación:

- La regla **28** es *redundante* con la regla **47**:
  28: -A INPUT -s 157.96.119.0/24 -j DROP
  47: -A INPUT -s 157.96.119.0/24 -j DROP
- La regla **33** es *inconsistente* con las reglas **74**, **75**, **77** y **78**:
  33: -A INPUT -s 157.96.138.0/24 -j DROP
  74: -A INPUT -s 157.96.138.138/32 -p tcp -m tcp –dport 5900 -j ACCEPT
  75: -A INPUT -s 157.96.138.138/32 -p tcp -m tcp –dport 5166 -j ACCEPT
  77: -A INPUT -s 157.96.138.101/32 -p tcp -m tcp –dport 5900 -j ACCEPT
  78: -A INPUT -s 157.96.138.101/32 -p tcp -m tcp –dport 5166 -j ACCEPT
- La regla **33** es *redundante* con las reglas **76** y las **79** a **84**:
  33: -A INPUT -s 157.96.138.0/24 -j DROP
  76: -A INPUT -s 157.96.138.138/32 -j DROP
  79: -A INPUT -s 157.96.138.101/32 -j DROP
  80: -A INPUT -s 157.96.138.80/32 -j DROP
  81: -A INPUT -s 157.96.138.82/32 -j DROP
  82: -A INPUT -s 157.96.138.234/32 -j DROP
  83: -A INPUT -s 157.96.138.235/32 -j DROP
  84: -A INPUT -s 157.96.138.236/32 -j DROP

La herramienta resolvió estos conflictos respetando la regla de mayor prioridad por defecto. Consideramos que estos resultados muestran la capacidad de la herramienta para mejorar la calidad de las políticas de firewall, eliminando errores de configuración de forma automatizada.

3.2 Discusión

La transformación del modelo FDD a un conjunto de reglas implica, en ciertos casos, que los campos de filtrado con conjuntos grandes de elementos deban dividirse en varios subconjuntos más pequeños. Este proceso puede generar, a partir de un conjunto inicial, un número mayor de reglas que las iniciales. Dado que el número de reglas no siempre disminuye e incluso puede aumentar, se puede concluir que no se cumple con la propiedad de compacidad en este enfoque.

Se han identificado posibles soluciones para mitigar este fenómeno, aunque ninguna de ellas ha sido implementada:

1. Las reglas cuyas direcciones ip son la sumatoria de alguna cantidad de redes o direcciones permanecen como una sola regla hasta el momento de hacer la transformación a la sintaxis de salida. Dado que iptables y otros sistemas no permiten especificar conjuntos de direcciones o redes en una misma regla, esta tiene que dividirse en reglas múltiples. Podrían identificarse estos casos e implementar una herramienta que en lugar de resolver el problema dividiendo la regla, la exprese como el caso general y su excepción, obteniendo una salida mucho más acotada.

2. En ciertas ocasiones, debido a las reglas expresadas en el firewall inicial o a cambios en el FDD, obtenemos nodos cuya decisión se bifurca en diferentes caminos que sin embargo podrían unificarse en uno solo. Existe la posibilidad de implementar un algoritmo que verifique la igualdad entre nodos adyacentes provenientes del mismo nodo superior y los unifique en caso de igualdad.

4 Conclusiones

La aplicación desarrollada resultó efectiva para mejorar las propiedades de completitud y consistencia. No se logró cumplir la propiedad de compacidad. Esto se debe a que el número final de reglas puede ser mayor al número inicial.

Consideramos que esta herramienta mejora dos de los aspectos cruciales para la gestión de las políticas de firewall:

- **Gestión gráfica de políticas de firewall:** Las primeras pruebas de representación de políticas de firewall como árbol de decisiones resultaron confusas a la lectura directa. Luego, cuando se agregó la funcionalidad de filtrado, mejoró su legibilidad. Con esto queremos decir que el gráfico permitió identificar de manera sencilla el comportamiento de la política.
- **Adición directa de reglas:** Con solo ingresar una instrucción a la aplicación es posible modificar el FDD y por lo tanto lograr un nuevo comportamiento. Esto evita, al administrador del firewall, tener que identificar la posición que la nueva regla debería ocupar en el conjunto de reglas.

En resumen, la herramienta permite tomar un set de reglas de firewall en funcionamiento, ingresarlo mediante un archivo, llevar a cabo su procesamiento y transformación al modelo de FDD, optimizarlo, representarlo gráficamente, editarlo y finalmente reconvertirlo a su formato original.

5 Referencias

[1] Fei Chen et al. “First step towards automatic correction of firewall policy faults”. In: *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 7.2 (2012), pp. 1–24.

[2] Thawatchai Chomsiri, Xiangjian He, and Priyadarsi Nanda. “Limitation of listed-rule firewall and the design of tree-rule firewall”. In: *Internet and Distributed Computing Systems: 5th International Conference, IDCS 2012, Wuyishan, Fujian, China, November 21-23, 2012. Proceedings 5*. Springer. 2012, pp. 275–287.

[3] Mohamed G Gouda and Alex X Liu. “Structured firewall design”. In: *Computer networks* 51.4 (2007), pp. 1106–1120.

[4] Ehab S Al-Shaer and Hazem H Hamed. “Modeling and management of firewall policies”. In: *IEEE Transactions on network and service management* 1.1 (2008), pp. 2–10.

[5] R.L. Ziegler and C.B. Constantine. *Linux Firewalls*. CampusPress référence. New Riders, 2002. Chap. 3 - iptables: The Linux Firewall Administration Program. ISBN: 9780735710993. URL: `https://books.google.com.ar/books?id=rIWkyUYBsCwC`.