Congr. Int. en Ing. Electrónica. Mem. ELECTRO, Vol. 40, pp. 67 - 72, Oct 2018, Chihuahua, Chih. México
http://electro.itchihuahua.edu.mx/memorias_electro/MemoriaElectro2018.zip
ISSN 1405-2172

SIMULADOR DE PUERTOS Y SERVICIOS DE RED EN UN SISTEMA EMBEBIDO

Vega Luna José Ignacio, Lagos Acosta Mario Alberto, Salgado Guzmán Gerardo, Sánchez Rangel Francisco Javier, Quiroz Rodríguez Alfredo
Universidad Autónoma Metropolitana-Azcapotzalco, Área de Sistemas Digitales, Dpto. de Electrónica.
Av. San Pablo 180, Col. Reynosa, C.P. 02200, Cd. de México.
vlji@correo.azc.uam.mx

RESUMEN.

En este trabajo se describe el diseño de un dispositivo auxiliar para pruebas de penetración en redes TCP/IP, en específico para la realización de análisis de puertos TCP y servicios de red con fines de entrenamiento en una red local sin comprometer la seguridad de equipos en producción. Utilizando sistema embebido Raspberry Pi 3 como hardware y Kali Linux como sistema operativo, se programaron en lenguaje Python rutinas para la apertura y escucha en puertos TCP específicos. Se seleccionan los puertos en escucha y como responderán a peticiones solicitadas, cuando se les solicita remotamente el acceso responden como se programó. La respuesta al análisis remoto se valida con la herramienta nmap. En el presente trabajo se simularon los servicios HTTP, SMTP, POP3, FTP, DNS, IRC, IMAP, Syslog, SNMP y DHCP, con la respuesta asociada de servicios Windows o Linux según sea seleccionado.

Palabras Clave: Kali Linux, nmap, prueba de penetración, Raspberry Pi 3, Simulación de red.

ABSTRACT.

This paper describes the design of an auxiliary device for penetration tests in TCP / IP networks, specifically for the analysis of TCP ports and network services for training purposes in a local network without compromising the security of equipment in production. Using Raspberry Pi 3 embedded system as hardware and Kali Linux as operating system, routines for opening and listening on specific TCP ports were programmed in Python language. The listening ports are selected and how they will respond to requested requests, when they are remotely requested access they respond as programmed. The response to remote analysis is validated with the nmap tool. In the present work, the HTTP, SMTP, POP3, FTP, DNS, IRC, IMAP, Syslog, SNMP and DHCP services were simulated, with the associated response of Windows or Linux services as selected.

Keywords: Kali Linux, nmap, network simulation, pen testing, Raspberry Pi 3.

1. INTRODUCCIÓN

La mayoría de las aplicaciones utilizadas a diario requieren conectarse a la Internet para usar servicios y aplicaciones que se ejecutan en servidores, esto conlleva la apertura de puertos de red con el fin de establecer una comunicación haciendo uso de los protocolos de la capa de transporte del modelo OSI TCP o UDP, que son los más utilizados [1]. También implica un riesgo o peligro, debido a que se puede efectuar un ataque usando la vulnerabilidad de algún puerto de red activo en el dispositivo. Debido a esto, todo el tiempo los dispositivos están en constante apertura y cierre de puertos de forma transparente, sin percatarse del posible peligro al que se está expuesto. Por ello, es importante realizar un análisis de los puertos para identificar estos peligros, ser proactivo y tomar acciones al respecto.

Existen herramientas para descubrir los puertos de red abiertos, las cuales realizan preguntas al sistema objetivo, esperando una respuesta acorde, lo cual se conoce como sondeo de puertos. Existen diferentes técnicas de este tipo tales como el sondeo TCP SYN, el sondeo TCP connect (), el sondeo UDP, entre otras. Los puertos tienen asociados servicios, por ejemplo, el puerto 67 con el servicio de DHCP que es un proceso ejecutándose en el equipo alojado. Dicho de otra forma, todo el tráfico perteneciente a este servicio se recibirá y enviará por el puerto antes mencionado. Un puerto es la interfaz utilizada para establecer una comunicación a través de la red. Existen puertos denominados bien conocidos, que son reservados y utilizados por el sistema operativo, son los puertos inferiores al 1024. También se encuentran los puertos registrados, los cuales pueden ser usados por cualquier aplicación y están en el rango de 1024 a 49151. Además de los anteriores, existen puertos dinámicos o privados, estos no tienen ningún propósito específico, y son los comprendidos entre el 49152 al 65535. El barrido de puertos es la acción en la cual se analiza el estado de los puertos de un dispositivo conectado a la red a través de herramientas diseñadas para este propósito [2]. El objetivo es determinar los servicios bien conocidos que ejecuta el dispositivo, así como las posibles vulnerabilidades de acuerdo con los puertos identificados. La utilidad de software libre más común para explorar, administrar y auditar la seguridad de redes es NMAP (Network Mapper). Algunas de las funciones de NMAP es reportar nodos de red en línea, puertos abiertos, los servicios y aplicaciones que tienen en ejecución. Las características del análisis dependen de las opciones seleccionadas para el análisis. El principal fin del análisis es la identificación de los puertos y los servicios asociados, que pueden reportar, puerto en escucha, puerto abierto, puerto cerrado y puertos en filtrados como se muestra en la Figura 1. Para realizar un reporte de puertos localmente en el mismo equipo, comúnmente se ejecuta el comando del sistema operativo *netstat*.

```
nmap 192.168.15.3

Starting Nmap 7.60 ( https://nmap.org )
de verano central (México)
Nmap scan report for 192.168.15.3
Host is up (0.0063s latency).
Not shown: 994 filtered ports
PORT     STATE  SERVICE
17/tcp   closed qotd
256/tcp  closed fw1-secureremote
264/tcp  closed bgmp
443/tcp  closed https
1080/tcp closed socks
1723/tcp closed pptp

Nmap done: 1 IP address (1 host up)
```

Fig. 1. Análisis con la herramienta nmap

Utilizando las diferentes opciones de *netstat* es una muy buena guía para determinar los puertos en escucha, las conexiones activas e incluso las aplicaciones asociadas a éstos como se indica en la Figura 2.

```
C:\>netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1031           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1311           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:2002           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3389           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5800           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5881           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5900           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5981           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8000           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:47001          0.0.0.0:0              LISTENING
  TCP    127.0.0.1:1034         0.0.0.0:0              LISTENING
  TCP    127.0.0.1:2002         127.0.0.1:65032        ESTABLISHED
  TCP    127.0.0.1:65032        127.0.0.1:2002         ESTABLISHED
  TCP    192.168.42.124:139     0.0.0.0:0              LISTENING
  TCP    192.168.42.124:3818    192.168.42.214:80      ESTABLISHED
  TCP    192.168.42.124:24756   192.168.42.205:80      ESTABLISHED
  TCP    192.168.42.124:27817   192.168.42.203:80      SYN_SENT
  TCP    192.168.42.124:27818   192.168.42.206:80      SYN_SENT
  TCP    192.168.42.124:34440   192.168.42.213:80      ESTABLISHED
  TCP    192.168.42.124:60248   64.94.18.156:443       ESTABLISHED
  TCP    192.168.42.124:65065   192.168.42.209:80      ESTABLISHED
  TCP    192.168.42.124:65067   192.168.42.210:80      ESTABLISHED
  TCP    192.168.42.124:65068   192.168.42.208:80      ESTABLISHED
  TCP    192.168.42.124:65072   192.168.42.204:80      ESTABLISHED
  TCP    192.168.42.124:65122   192.168.42.242:80      CLOSE_WAIT
  UDP    0.0.0.0:161            *:*
  UDP    0.0.0.0:445            *:*
  UDP    0.0.0.0:500            *:*
  UDP    0.0.0.0:2148           *:*
  UDP    0.0.0.0:4500           *:*
  UDP    0.0.0.0:27822          *:*
  UDP    127.0.0.1:123          *:*
  UDP    192.168.42.124:123     *:*
  UDP    192.168.42.124:137     *:*
  UDP    192.168.42.124:138     *:*
  UDP    192.168.42.124:42450   *:*
  UDP    192.168.42.124:42451   *:*
```

Fig. 2. Resultados de ejecución de netstat

Actualmente, existen diversas herramientas software para dar de alta servicios de red específicos, que inician procesos asociados. Por ejemplo, el protocolo HTTP (Hypertext Transfer Protocol) puede ser iniciado con la aplicación conocida como Apache. Estas aplicaciones consumen una cantidad importante de recursos hardware para ser ejecutada. Con las herramientas mencionadas solo se obtiene parte de la información que transita por la red entre clientes y servidores. Para obtener toda la información es necesario una herramienta de escucha de tráfico que comúnmente se llamada sniffer o analizador de protocolos. Los sniffer se encargan de escuchar el tráfico en las o las interfaces de red y almacenar los resultados de la escucha. La herramienta que es más común para realizar esta tarea es la llamada Wireshark. Es una herramienta multiplataforma de análisis de red. Trabaja similar a otras de su tipo como Windump, TCPDump ó dsniff, pero a diferencia de éstas últimas, muestra la información capturada gráficamente, de forma más amigable y entendible, como se muestra en la Figura [3].

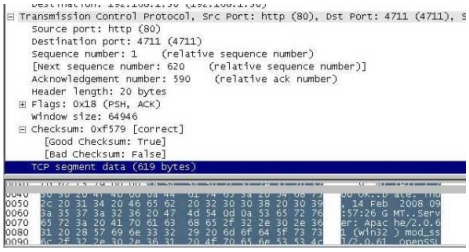

Fig. 3. Ventana de resultados utilizando Wireshark

Los análisis descritos que se realizan con las diferentes herramientas, son parte de lo nombrado pruebas de penetración. Al realizar algunas de estas pruebas se utilizan los recursos de equipos remotos y se puede influir en el rendimiento de los equipos. Así mismo existen algunos análisis remotos que pueden ser intrusivos y con esto causar una negación de servicios, lo que afecta la productividad en los usuarios. Un caso común de la negación de servicios producida por análisis remotos de equipos, es realizar el análisis de una máquina que tiene habilitada una herramienta de seguridad, como puede ser un IPS (Sistema de Prevención a Intrusiones), y el IPS bloquea servicios y/o puertos después de intentos de accesos o consultas que son considerados como sospechosos y se identifican, para solicitar un bloqueo del puerto precautoriamente. Al fenómeno del bloqueo se considera como una negación de servicio [4]. Comúnmente al capacitar a personal para tareas de seguridad en redes, las prácticas de análisis por barrido no se realizan usando equipos productivos, ya que podrían ocasionar más problemas que beneficios [5]. Los sistemas de simulación de servicios de red, son de gran utilidad para el entrenamiento del personal dedicado a las tareas de seguridad. Existen herramientas que se ejecutan en máquinas virtuales, pero para obtener experiencia en estas tareas el realizar las pruebas con hardware nos ayuda a comprender más los conceptos de seguridad de redes.

El diseño de una herramienta que simule múltiples servicios de red, ayuda a mitigar el consumo excesivo de recursos de una red con equipos productivos, ya que evita la instalación de aplicaciones solo con fines de evaluación y entrenamiento. En este trabajo se diseña un sistema donde el enfoque principal es la capacitación del personal de seguridad en redes. Al diseñar un hardware independiente, donde fácilmente se pueda adecuar las características de funcionamiento en la red, como la apertura de puertos con servicios que no están implementados y que solo responden a barridos como habitualmente se responde en los sistemas originales, como son HTTP, SMTP, POP3,

FTP, DNS, IRC, IMAP, Syslog, SNMP y DHCP. Todo esto reduce los tiempos de capacitación y acondicionamiento de sistemas para este fin [6].

2. DESARROLLO

En la realización de este trabajo la metodología seguida fue dividir el diseño en tres etapas. La primera etapa fue la instalación del ambiente operativo en la tarjeta Raspberry Pi 3, la segunda etapa consistió en realizar la captura del barrido y la tercera etapa fue el desarrollo de la aplicación para la selección de simulación.

2.1. Instalación del ambiente operativo en la tarjeta Raspberry.

En el sitio offensive-security dedicado a implementar a diferentes plataformas de hardware el sistema operativo Kali Linux. Se descargó el sistema operativo Kali Linux en su versión ARM correspondiente al modelo del Raspberry Pi de la dirección https://www.offensive-security.com/kali-linux-arm-images. Para realizar la instalación del sistema operativo y arrancar la tarjeta Raspberry, se utilizó el programa llamado Win32 Disk Imager, con el cual se creó la imagen de Kali Linux en una tarjeta de memoria SD [7]. Posteriormente, se conectaron los siguientes dispositivos periféricos a la tarjeta Raspberry: teclado, mouse y monitor HDMI. Se insertó la tarjeta SD y se encendió la tarjeta Raspberry Pi 3. Se actualizaron parches y herramientas de red ejecutando desde una sesión de terminal los comandos: *apt-get update* y *apt-get upgrade*. En sistema diseñado no es necesario implementar alguna conexión física que no esté implementada en el kit de desarrollo de nuestro sistema Raspberry embebido, solo usando los puertos SD, HDMI y Ethernet. En la puesta a punto es importante no arrancar servicios de red innecesarios, ya que esto podría causar conflictos con los análisis de red que se soliciten remotamente. El único servicio arrancado con fines administrativos fue ssh, para poder conectar al sistema embebido y configurar utilizando una terminal remota. Los parámetros de red dirección IP, máscara de red y puerta de enlace, se establecen de acuerdo a la red donde este sistema trabaje.

2.2. Captura de barrido.

En esta etapa se lleva a cabo el barrido con sistemas que tienen servicios y puertos en operación. La finalidad de esta etapa del sistema es capturar las respuestas que se obtienen con cada uno de los servicios, para poder replicar las respuestas a cada una de las peticiones en la simulación de los servicios. El procedimiento de captura consiste de las siguientes tareas: Instalar el servicio en un equipo Linux, Windows o MacOs, realizar el barrido al puerto determinado del equipo servidor ejecutando remotamente el comando *nmap*, capturar las características de la petición del cliente al servidor almacenando la información en un registro de petición por puerto y capturar la respuesta del servidor almacenando en un registro de respuestas del servidor por puerto. En la Figura 4 se muestra el procedimiento de captura.

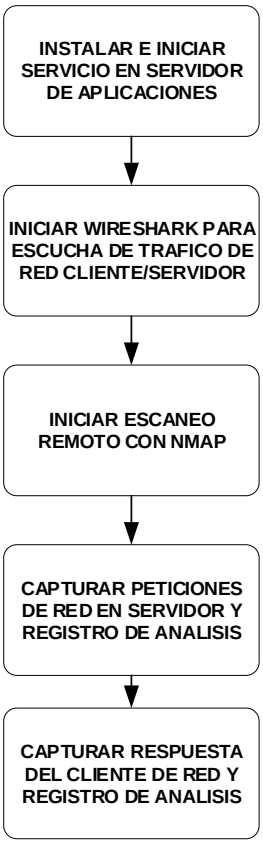

Fig. 4. Procedimiento de captura de protocolos de red

En base a los registros obtenidos, se programaron las rutinas de escucha y respuesta a las peticiones del sistema Raspberry con Linux. Al realizar las capturas del análisis de puertos, éstas se almacenan en una base de datos conjuntamente con las características del servicio que están simulando en cada análisis. Con los datos obtenidos se pueden simular los servicios de un sistema Windows en un sistema Linux (Raspberry Pi). Con la programación realizada, como por ejemplo al realizar el análisis remoto al puerto 80 del protocolo HTTP, se obtiene la huella de la aplicación asociada, también llamada fingerprint, como se indica en la Figura 5.

2.3. Desarrollo de la aplicación para la selección de simulación.

Las aplicaciones de escucha consisten de cuatro módulos: el módulo de lectura de parámetros de operación, el módulo de inicialización de puertos, el módulo de respuesta a peticiones y el módulo de resultados.

2.3.1 El módulo de lectura de parámetros de operación.

Este módulo inicializa el funcionamiento de puertos, aplicaciones y sus servicios asociados a cada puerto. Como ejemplo, el puerto 80 se utiliza principalmente para servicios del protocolo HTTP en aplicaciones WEB y existen múltiples asociaciones a este puerto, que estarán determinadas a la configuración con la que se inicializo el sistema.

```
def puerto_http():
        print "Puerto HTTP"
        banner = "HTTP/1.1 200 OK\x0d\x0a\
Date: Tue, 22 May 2018 17:51:23 GMT\x0d\x0a\
Server: Apache/2.4.25 (Debian)\x0d\x0a\
Allow: OPTIONS,HEAD,HEAD,GET,HEAD,POST\x0d\x0a\
Content-Length: 0\x0d\x0a\
Connection: close\x0d\x0a\
Content-Type: text/html\x0d\x0a\
\x0d\x0a"
        TCP_IP = ip_address
        TCP_PORT = 80
        BUFFER_SIZE = 1024

        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.bind((TCP_IP, TCP_PORT))
        s.listen(1)

        conn, addr = s.accept()
        print 'HTTP Connection address: ', addr

        while True:
                data = conn.recv(BUFFER_SIZE)
                if not data: break
                #print "Received data HTTP:", data
                conn.send(banner)
                archivo1.write("----HTTP fingerprint---- \x0d\x0a" + str(banner))
        conn.close()
        print 'Termino HTTP'
        return
```

Fig. 5. Programación de huella del servicio asociado al puerto 80 HTTP

En el archivo de texto usado para inicializar los puertos cada renglón indica el puerto que se desea abrir para la simulación. Los parámetros con los que opera la aplicación se encuentran en el archivo llamado *ptscfg.txt*, contenido en la memoria SD para describir las características de la simulación a realizar. El archivo tiene la sintaxis que se muestra en la Figura 6.

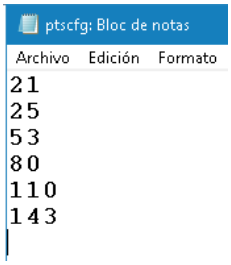

Fig. 6. Archivo de configuración ptscfg.txt

2.3.2 El módulo de inicialización de puertos.

En este módulo se programan los sockets correspondientes a cada uno de los puertos que se habilitan como abiertos o en escucha. Los puertos están en espera a un llamado para poder atender a la petición. Esto es conforme a la configuración obtenida en el módulo anterior. Con el archivo de configuración habilita en escucha los puertos 21, 25, 53, 80, 110 y 143, las rutinas de programación dedicadas a abrir los puertos y serán algo repetitivas, por tener una lógica muy similar.

2.3.3 El módulo de respuesta a peticiones.

Este módulo consiste en programar la respuesta a cada una de las peticiones remotas como deben responder típicamente los servicios simulados. En el caso del puerto 80, al realizar un barrido responde con la huella del servicio Apache para remotamente identificar las características del servicio en operación. Estas huellas inicialmente se capturaron con la aplicación de escucha de tráfico Wireshark, realizando una petición remota con *nmap*, al analizar un sistema que realmente tiene en ejecución la aplicación. El diagrama de flujo de la aplicación desarrollada en el lenguaje Python se muestra en la Figura 7.

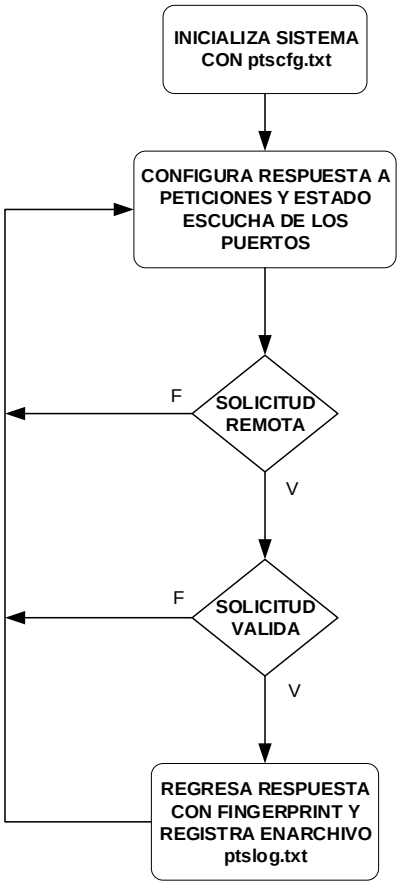

Fig. 7. Diagrama de flujo de aplicación principal

2.3.4 El módulo de resultados.

Los resultados obtenidos en las solicitudes se almacenan en un archivo llamado ptslog.txt en el que se registran las solicitudes y respuestas exitosas a cada uno de los puertos configurados. La sintaxis del archivo de bitácora muestra en cada renglón el puerto consultado cada vez que tuvo una respuesta exitosa, solo se registran las peticiones a puertos configurados, como se muestra en la Figura 8.

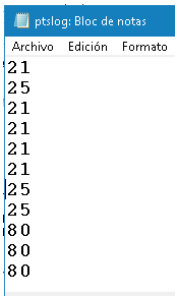

Fig. 8. Archivo de registro de peticiones a los puertos

3. RESULTADOS

Al realizar pruebas del sistema, se utilizó la configuración de inicialización con los puertos 5, 21, 25, 53, 67, 80, 110, 143, 161, 194 y 514. De estos puertos fue capturado previamente su función y respuestas con *nmap*. Se ejecutó la rutina la cual abre los puertos en escucha de la que se obtienen los resultados y al realizar la comprobación de puertos utilizando localmente en el simulador el comando *netstat*, comprobamos que localmente en el sistema Rasberry Pi se encuentran en escucha dichos puertos como se muestra e la Figura 9.

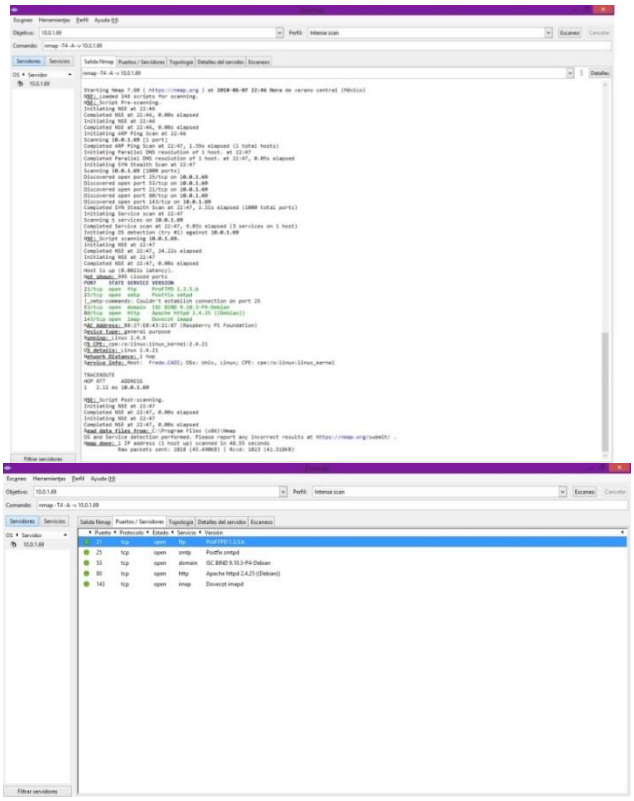

Fig. 9. Puertos en escucha

El siguiente paso fue realizar un barrido intenso utilizando *nmap* desde un cliente en la red local y con destino la maquina servidor.

Al final del barrido se obtuvo el resumen de resultados con *nmap*, donde nos muestra que los puestos que en el archivo de configuración inicial se solicita la apertura, realmente al usar la herramienta de acceso remoto se observan en escucha.

En La consola del servidor se obtiene el listado de las peticiones solicitadas como se indica en la Figura 10, igualmente se genera el archivo donde se registra la bitácora *ptslog.txt*.

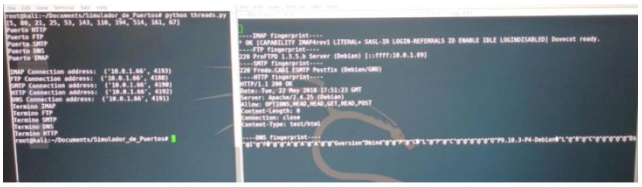

Fig. 10. Consola de resultados en el servidor de la aplicación

4. CONCLUSIONES

Se alcanzaron los objetivos satisfactoriamente. El principal fin de este trabajo es el mostrar las diferentes firmas que nos responden los servicios alojados en un servidor sin instruir en el funcionamiento en un equipo productivo. En una práctica de laboratorio de seguridad es de gran utilidad poder hacer cambios de parámetros en los equipos servidores, como la apertura de puertos y su análisis, así como con qué servicios están asociados, sin necesidad de realizar instalaciones tardadas en el sistema que se analiza, además fácilmente se pueden simular respuestas de servicios que deberían estar alojados en sistemas Windows, utilizando un servicio Linux. Cabe mencionar que se obtuvo un trabajo el cual puede crecer y modificarse según las necesidades del uso. En este caso solo se realizaron simulaciones para protocolos que responden a las peticiones con códigos alfanuméricos, ya que es más descriptivo. Las adecuaciones que se pueden implementar a futuros trabajos al sistema, es la automatización de las capturas de los escaneos. La automatización constara en realizar un sistema específico a capturas de huellas de los diferentes servicios que se pueden instalar en un servicio.

5. REFERENCIAS

[1] N. Baldo, Validation of the ns-3 ieee 802.11 model using the extreme testbed. In Proceedings of SIMUTools Conference, 2010, March 2010.

[2] A. K. Kyaw, Pi-IDS: evaluation of open-source intrusion detection systems on Raspberry Pi 2. 2015 Second International Conference on Information Security and Cyber Forensics (InfoSec). Cape Town, Nov. 2015. Pages: 165-170.

[3] E. Weing, A performance comparison of recent network simulators. In Proceedings of the IEEE International Conference on Communications 2009 (ICC 2009), Dresden, Germany, 2009.

[4] C. Zena, Penetration testing: Concepts, attack methods, and defense strategies. 2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT). New York, April 29, 2016. Pages: 89-94.

[5] G. J. N. Mehtre, Dynamic IPv6 activation based defense for IPv6 router advertisement flooding (DoS) attack. 2014 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC). Coimbatore, Dec. 2014, Pages: 1–5.

[6] A. Durrani, Analysis and prevention of vulnerabilities in cloud applications. 2014 Conference on Information Assurance and Cyber Security (CIACS). Rawalpindi, June, 2014. Pages: 43-46.

[7] Raspberry Pi Foundation, "Installing operating system images using Windows" https://www.raspberrypi.org/documentation/installation/installing-images/windows.md, Dic. 2016.