udp FACULTAD DE INGENIERÍA

Taller de redes y servicios

Análisis Informático y de Telecomunicaciones de Protocolo de Aplicación de red

Integrantes: Cristián Villavicencio
Profesor: Nicolás Boettcher
Ayudante: Miguel Contreras

Índice

1. Introducción

El presente informe detalla el estudio y análisis del protocolo IRC, con el fin de lograr comprender el funcionamiento de este, así como su comportamiento, para poder lograr esto se hará uso de un sistema cliente/servidor que utilice dicho protocolo, este sistema deberá ser implementado utilizando contenedores, instalando todas las dependencias necesarias así como el software mediante el código fuente de estos, una vez realizado esto se procederá a capturar el trafico generado por este sistema cliente/servidor, buscando 10 tipos distintos asociados al protocolo, finalmente se buscara intentar modificar este trafico mediante configuración del software y buscar 5 vulnerabilidades que posea.

2. Protocolo IRC

Internet Relay Chat en su sigla IRC, es un protocolo de comunicación de texto en tiempo real, este permite mensajería entre dos o mas miembros mediante el esquema cliente/servidor, es decir se encuentra una conexión con el servidor y no entre los usuarios de forma directa, fue creado en 1988 por Jarkko Oikarinen, inspirado en Bitnet.

Dentro de los servidores y clientes IRC podremos encontrar una serie de elementos, como por ejemplo los canales, los operadores (moderadores) los cuales facilitaran el esquema de comunicación, organización y jerarquía para poder realizar una conexión mas eficaz y segura, se detallaran las mas esenciales para el estudio del trafico generado:

- **Connection Registration**
    1. Nick massage: comando utilizado para asignar/modificar nick de usuario.
    2. User message: se usa al principio de cada conexión para indicar el nombre de usuario, de host y servidor y el nombre real del nuevo usuario. Se usa también en la comunicación entre servidores para indicar que un nuevo usuario llega a la red de IRC.
    3. Quit: Indica la sesión finalizada de un usuario,
- **Channel operations:**
    1. Join message: El comando JOIN lo usa un usuario para solicitar comenzar a escuchar el canal específico.
    2. Kick command: El comando KICK se puede utilizar para solicitar la expulsión forzada de un usuario de un canal.
    3. List message: Sirve para ver la lista de canales y sus tópicos.
- **Server queries and commands:**
    1. Time message: El comando de hora se utiliza para consultar la hora local de la especificada servidor.
    2. Connect message: El comando CONNECT se puede utilizar para solicitar a un servidor que intente establezca una nueva conexión a otro servidor inmediatamente.
    3. Info command: El comando CONNECT se puede utilizar para solicitar un servidor que intente establezca una nueva conexión a otro servidor inmediatamente.

- **User based queries:**

    1. Who query: Comando que recibe de parámetro la mascara, para realizar una petición de recibir información.
    2. Whois query: Similar al anterior, pero en base a un usuario en concreto.
    3. Whowas: Este funciona con los nick y ver la información asociada a este.

- **Miscellaneous messages:**

    1. Kill message: El comando KILL se usa para hacer que una conexión cliente-servidor sea cerrado por el servidor que tiene la conexión real.
    2. Ping message: El comando PING se utiliza para probar la presencia de un cliente activo o servidor en el otro extremo de la conexión.
    3. Pong message: Es la respuesta generada por Ping
    4. Error: Mensaje enviado por el servidor cuando encuentra fallas en este.

Para ver de forma extensa las propiedades de este protocolo visitar los enlaces en la referencia[11][13][14]

3. Softwares

En el presente apartado se detallara la configuración de los softwares y dependencias necesarias para poder realizar el análisis, se repartirá en 3 sub-apartados, el primero indicara todo software necesario para poder realizar la configuración esencial que se utilizo para el análisis, así como los softwares usados. Posterior a esto se detallara la configuración del servidor y el cliente.

3.1. Ubuntu

Para el desarrollo del trabajo se utilizara la distribución de Linux Ubuntu[1] 20.04 lts, un sistema operativo de código abierto, en este usaremos esencialmente la terminal debido a la facilidad de instalar paquetes y software mediante código fuente.

Imagen I: Logo Ubuntu

3.2. Wireshark

Wireshark[2] es un software que nos permitirá analizar el flujo de los paquetes de distintos protocolos que se comunican en una red, con el y los filtros podremos realizar el análisis de forma mas detallada.

Imagen II: Logo Wireshark

3.3. Docker

Docker[3] es un software que nos permitirá desplegar las aplicaciones mediante contenedores, instalando solamente los requerimientos elementales de cada software, para poder optimizar su uso, así como facilitar el análisis del protocolo.

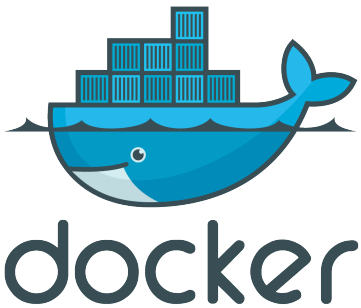

Imagen III: Logo Docker

3.4. Git

Git[4] es un software de control de versiones, con el podremos administrar de forma mas eficaz la forma en que modificamos archivos, además se usara la plataforma Github[5], de esta se obtuvo el código fuente del cliente y el servidor. En esta se subirá un repositorio con el código fuente de la tarea.

Imagen IV: Logo Git y Github

3.5. Servidor: Oragono

Oragono[6] es un servidor de IRC, escrito en GO, lo usaremos para realizar el análisis, instalándolo por código fuente que se encuentra en un repositorio github[7], para el análisis usaremos la versión 2.3

Imagen V: Logo Oragono

3.6. Cliente: Weechat

Weechat[8] es un cliente para IRC, escrito en C, lo usaremos para realizar el análisis, instalándolo por código fuente que se encuentra en un repositorio github[9], para el análisis usaremos la versión 2.9

Imagen VI: Logo Weechat

3.7. Configuración

Para poder realizar el trafico primero debemos instalar los softwares en un contenedor mediante el código fuente, para ello usaremos un Dockerfile, para cada uno, instalando las dependencias esenciales de los softwares, para consultar la configuración de estos dockerfile y ver como desplegar estos, visitar el repositorio IRC Analysis[10], en este existe un archivo readme, acompañado de una guía de como desplegar y un vídeo[12].

4. Análisis del trafico

4.1. Trafico 1:

Ping: Generado para ver estado del server

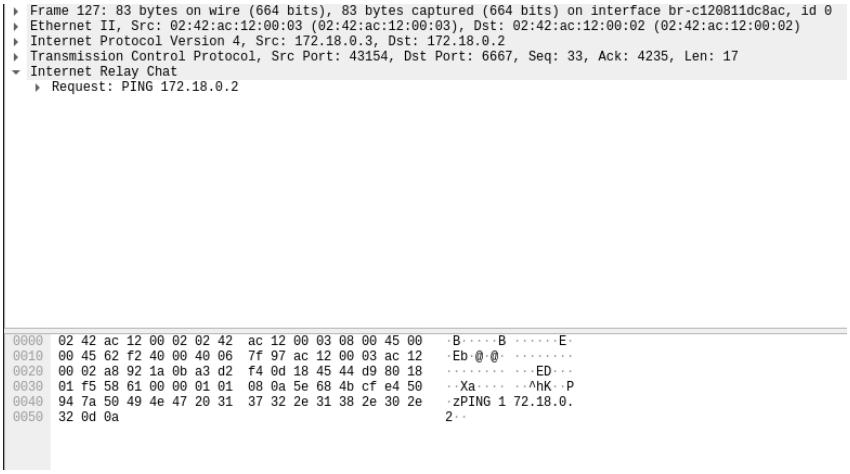

Imagen VI: Ping

4.2. Trafico 2:

Ping: Respuesta del ping.

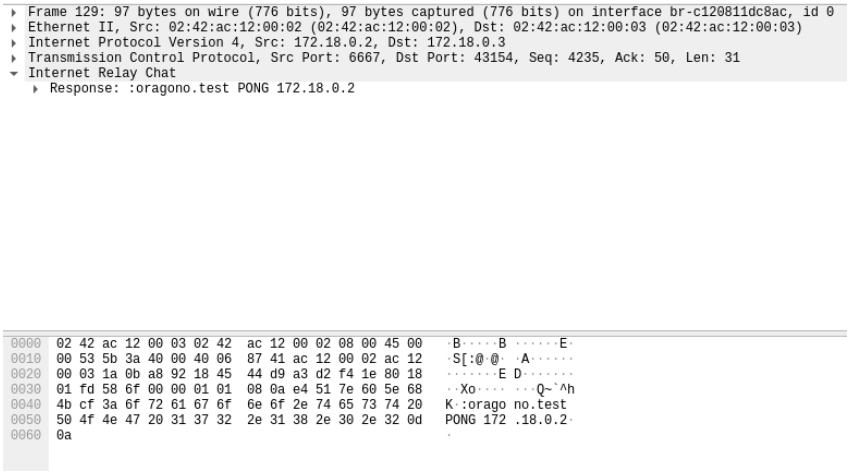

Imagen VII: Pong

4.3. Trafico 3:

User: Generado cuando un usuario se conecta a la red.

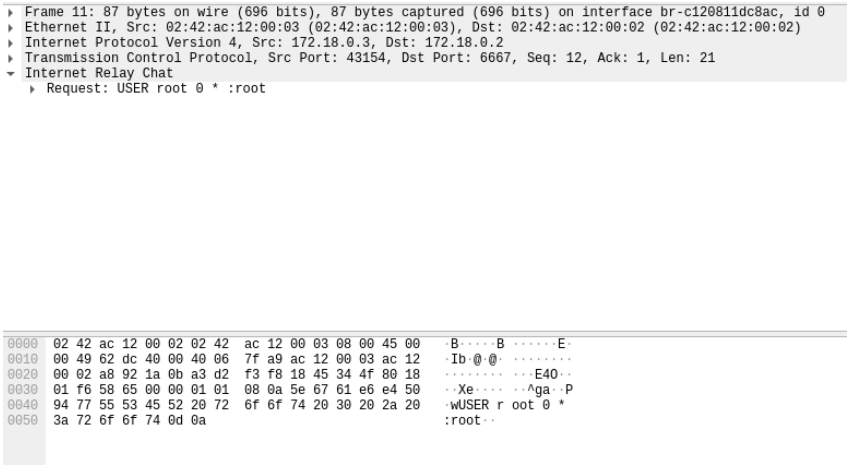

Imagen VII: User

4.4. Trafico 4:

Nick: Generado cuando usuario entra a la red, verifica nick

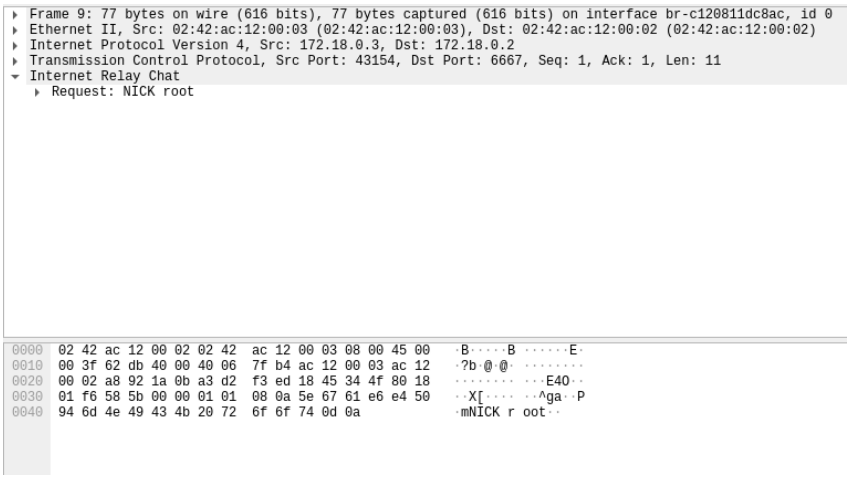

Imagen VIII: NIck

4.5. Trafico 5:

Join: Generado cuando usuario entra a un canal.

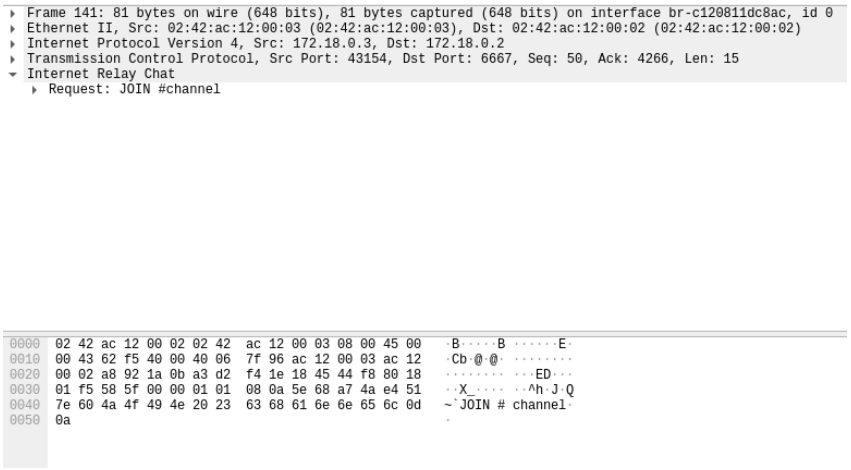

Imagen IX: Join

4.6. Trafico 6:

Part: Generado cuando usuario abandona canal.

```
▸ Frame 208: 94 bytes on wire (752 bits), 94 bytes captured (752 bits) on interface br-c120811dc8ac, id 0
▸ Ethernet II, Src: 02:42:ac:12:00:03 (02:42:ac:12:00:03), Dst: 02:42:ac:12:00:02 (02:42:ac:12:00:02)
▸ Internet Protocol Version 4, Src: 172.18.0.3, Dst: 172.18.0.2
▸ Transmission Control Protocol, Src Port: 43154, Dst Port: 6667, Seq: 178, Ack: 4630, Len: 28
▾ Internet Relay Chat
  ▸ Request: PART #channel :WeeChat 2.9
```

```
0000  02 42 ac 12 00 02 02 42  ac 12 00 03 08 00 45 00   ·B·····B ····· E·
0010  00 50 63 05 40 00 40 06  7f 79 ac 12 00 03 ac 12   ·Pc·@·@· ·y·····
0020  00 02 a8 92 1a 0b a3 d2  f4 9e 18 45 46 64 80 18   ········ ···EFd··
0030  01 f5 58 6c 00 00 01 01  08 0a 5e 6a 00 0e e4 52   ··Xl···· ··^j···R
0040  e2 ce 50 41 52 54 20 23  63 68 61 6e 6e 65 6c 20   ··PART # channel
0050  3a 57 65 65 43 68 61 74  20 32 2e 39 0d 0a         :WeeChat  2.9··
```

Imagen X: Part

```
▸ Frame 210: 124 bytes on wire (992 bits), 124 bytes captured (992 bits) on interface br-c120811dc8ac, id 0
▸ Ethernet II, Src: 02:42:ac:12:00:02 (02:42:ac:12:00:02), Dst: 02:42:ac:12:00:03 (02:42:ac:12:00:03)
▸ Internet Protocol Version 4, Src: 172.18.0.2, Dst: 172.18.0.3
▸ Transmission Control Protocol, Src Port: 6667, Dst Port: 43154, Seq: 4630, Ack: 206, Len: 58
▾ Internet Relay Chat
  ▸ Response: :root!~root@pjzui8bqt793s.irc PART #channel :WeeChat 2.9
```

```
0000  02 42 ac 12 00 03 02 42  ac 12 00 02 08 00 45 00   ·B·····B ····· E·
0010  00 6e 5b 4c 40 00 40 06  87 14 ac 12 00 02 ac 12   ·n[L@·@· ·····
0020  00 03 1a 0b a8 92 18 45  46 64 a3 d2 f4 ba 80 18   ·····E Fd·····
0030  01 fd 58 8a 00 00 01 01  08 0a e4 53 32 9f 5e 6a   ··X····· ···S2·^j
0040  00 0e 3a 72 6f 6f 74 21  7e 72 6f 6f 74 40 70 6a   ··:root! ~root@pj
0050  7a 75 69 38 62 71 74 37  39 33 73 2e 69 72 63 20   zui8bqt7 93s.irc
0060  50 41 52 54 20 23 63 68  61 6e 6e 65 6c 20 3a 57   PART #ch annel :W
0070  65 65 43 68 61 74 20 32  2e 39 0d 0a               eeChat 2 .9··
```

Imagen XI: Part 2

4.7. Trafico 7:

Privmsg: Generado cuando usuario chatea en canal.

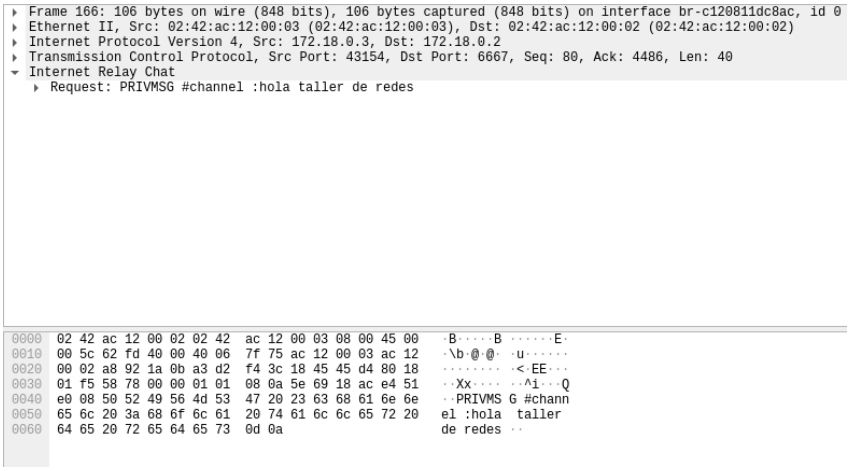

Imagen VI: Privmsg

4.8. Trafico 8:

Quit: Generado cuando usuario abandona la red.

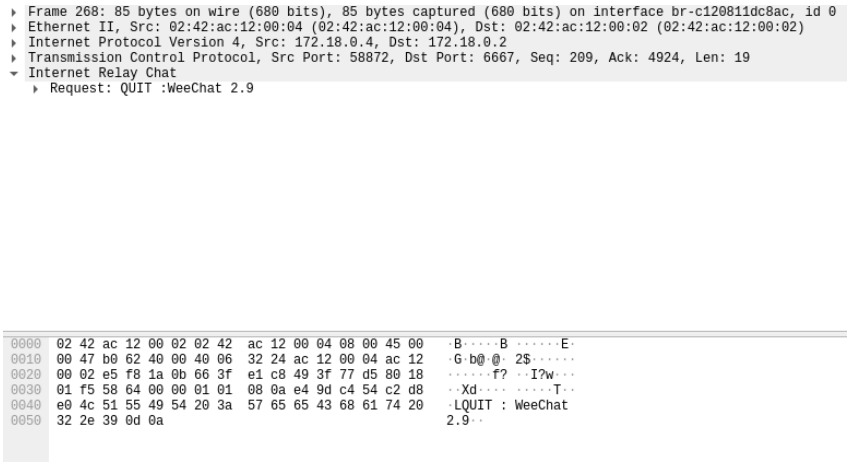

Imagen XII: Quit

4.9. Trafico 9:

Error: Generado cuando el servidor encuentra fallas.

```
▸ Frame 269: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on interface br-c120811dc8ac, id 0
▸ Ethernet II, Src: 02:42:ac:12:00:02 (02:42:ac:12:00:02), Dst: 02:42:ac:12:00:04 (02:42:ac:12:00:04)
▸ Internet Protocol Version 4, Src: 172.18.0.2, Dst: 172.18.0.4
▸ Transmission Control Protocol, Src Port: 6667, Dst Port: 58872, Seq: 4924, Ack: 228, Len: 82
▾ Internet Relay Chat
    ▸ Response: :root1!~root@i9qezr8uupwmi.irc QUIT :Quit: WeeChat 2.9
    ▸ Response: ERROR :Quit: WeeChat 2.9
```

```
0000  02 42 ac 12 00 04 02 42  ac 12 00 02 08 00 45 00   ·B·····B ····· E·
0010  00 86 83 e1 40 00 40 06  5e 66 ac 12 00 02 ac 12   ····@·@· ^f·····
0020  00 04 1a 0b e5 f8 49 3f  77 d5 66 3f e1 db 80 18   ·····I? w·f?····
0030  01 fd 58 a3 00 00 01 01  08 0a c2 d9 6a 7f e4 9d   ··X····· ····j···
0040  c4 54 3a 72 6f 6f 74 31  21 7e 72 6f 6f 74 40 69   ·T:root1 !~root@i
0050  39 71 65 7a 72 38 75 75  70 77 6d 69 2e 69 72 63   9qezr8uu pwmi.irc
0060  20 51 55 49 54 20 3a 51  75 69 74 3a 20 57 65 65    QUIT :Q uit: Wee
0070  43 68 61 74 20 32 2e 39  0d 0a 45 52 52 4f 52 20   Chat 2.9 ··ERROR
0080  3a 51 75 69 74 3a 20 57  65 65 43 68 61 74 20 32   :Quit: W eeChat 2
0090  2e 39 0d 0a                                       .9··
```

Imagen XIII: Error

4.10. Trafico 10:

Who: Generado por comando, para verificar datos de cierta mascara.

```
▸ Frame 231: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface br-c120811dc8ac, id 0
▸ Ethernet II, Src: 02:42:ac:12:00:04 (02:42:ac:12:00:04), Dst: 02:42:ac:12:00:02 (02:42:ac:12:00:02)
▸ Internet Protocol Version 4, Src: 172.18.0.4, Dst: 172.18.0.2
▸ Transmission Control Protocol, Src Port: 58872, Dst Port: 6667, Seq: 145, Ack: 4692, Len: 7
▾ Internet Relay Chat
    ▸ Request: WHO 0
```

```
0000  02 42 ac 12 00 02 02 42  ac 12 00 04 08 00 45 00   ·B·····B ····· E·
0010  00 3b b0 56 40 00 40 06  32 3c ac 12 00 04 ac 12   ·;·V@·@· 2<·····
0020  00 02 e5 f8 1a 0b 66 3f  e1 88 49 3f 76 ed 80 18   ·····f? ··I?v···
0030  01 f5 58 58 00 00 01 01  08 0a e4 9c df ba c2 d7   ··XX···· ········
0040  f8 8f 57 48 4f 20 30 0d  0a                       ··WHO 0· ·
```

Imagen XIV: Who

4.11. Trafico 11:

Whois: Generado por comando, podemos ver los datos añadidos antes.

```
▸ Frame 246: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface br-c120811dc8ac, id 0
▸ Ethernet II, Src: 02:42:ac:12:00:04 (02:42:ac:12:00:04), Dst: 02:42:ac:12:00:02 (02:42:ac:12:00:02)
▸ Internet Protocol Version 4, Src: 172.18.0.4, Dst: 172.18.0.2
▸ Transmission Control Protocol, Src Port: 58872, Dst Port: 6667, Seq: 197, Ack: 4830, Len: 12
▾ Internet Relay Chat
  ▸ Request: WHOIS list
```

```
0000  02 42 ac 12 00 02 02 42  ac 12 00 04 08 00 45 00   ·B·····B ····· E·
0010  00 40 b0 5d 40 00 40 06  32 30 ac 12 00 04 ac 12   ·@·]@·@· 20·····
0020  00 02 e5 f8 1a 0b 66 3f  e1 bc 49 3f 77 77 80 18   ·····f? ··I?ww··
0030  01 f5 58 5d 00 00 01 01  08 0a e4 9d 3a 21 c2 d8   ··X]···· ····:!··
0040  99 12 57 48 4f 49 53 20  6c 69 73 74 0d 0a         ··WHOIS  list··
```

Imagen XV: Whois

4.12. Trafico 12:

Notice: Generado con cierre abrupto del servidor, avisa sobre esta accion.

```
▸ Frame 284: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface br-c120811dc8ac, id 0
▸ Ethernet II, Src: 02:42:ac:12:00:02 (02:42:ac:12:00:02), Dst: 02:42:ac:12:00:03 (02:42:ac:12:00:03)
▸ Internet Protocol Version 4, Src: 172.18.0.2, Dst: 172.18.0.3
▸ Transmission Control Protocol, Src Port: 6667, Dst Port: 43154, Seq: 4828, Ack: 264, Len: 52
▾ Internet Relay Chat
  ▸ Response: :oragono.test NOTICE root :Server is shutting down
```

```
0000  02 42 ac 12 00 03 02 42  ac 12 00 02 08 00 45 00   ·B·····B ····· E·
0010  00 68 5b 58 40 00 40 06  87 0e ac 12 00 02 ac 12   ·h[X@·@· ·····
0020  00 03 1a 0b a8 92 18 45  47 2a a3 d2 f4 f4 80 18   ·····E G*·····
0030  01 fd 58 84 00 00 01 01  08 0a e4 55 58 92 5e 6b   ··X····· ···UX·^k
0040  f5 4f 3a 6f 72 61 67 6f  6e 6f 2e 74 65 73 74 20   ·O:orago no.test
0050  4e 4f 54 49 43 45 20 72  6f 6f 74 20 3a 53 65 72   NOTICE r oot :Ser
0060  76 65 72 20 69 73 20 73  68 75 74 74 69 6e 67 20   ver is s hutting
0070  64 6f 77 6e 0d 0a                                 down··
```

Imagen XVI: Notice

5. Vulnerabilidades

5.1. Vulnerabilidad 1:

El protocolo no escala suficientemente bien en proporciones grandes, pues como requisito debe ser que todos los servidores y clientes y se actualice de manera eficaz sus parámetros en caso de ser modificados.

5.2. Vulnerabilidad 2:

La colisión de los nick es un problema frecuente en este protocolo particular, pues al estar restringida la cantidad de caracteres y longitud, existe un espectro finito de ellos, por lo cual genera problemas en donde solo un usuario puede estar con actual nick, en caso de colisión existirá un KILL para ambos.

5.3. Vulnerabilidad 3:

Al conocer a detalle la información del trafico generado, se podría buscar forma de generar trafico "falso", para poder sobrecargar el server (DDoS) o realizar posibles acciones, dentro del server de forma de generar cambios en este.

5.4. Vulnerabilidad 4:

La jerarquía existente entre la conexión entre servidores es de tipo árbol, es decir que si una comunicación falla, podría generar una falla enorme en el sistema, pues podrían quedar muchos servidores incomunicados.

5.5. Vulnerabilidad 5:

Al ser un protocolo no cifrado, existe una gran vulnerabilidad para ser interceptado los paquetes que genera este y por ende posibles problemas en su seguridad.

6. Conclusiones

Gracias al trabajo se logro una cantidad de aprendizaje en distintas áreas, que están directamente relacionadas, por ejemplo desde la previa investigación de elegir un protocolo que nos facilitara el realizar el trabajo, así como poder investigar servicios de cliente/servidor relacionados a este, logrando también aprender nuevas tecnologías o mejorar en algunas, como docker, el uso de linux y también el análisis que podemos generar al estudiar la aplicación, de distintos ámbitos ya sea desde su documentación, código fuente y usos que esta tiene.

Se considera satisfactoria la experiencia, pero se buscara a futuro poder realizar un análisis mas profundo, para poder optimizar las pruebas, haciendo mas test y realizar un contenedor mas ligero.

7. Referencias

1. Ubuntu, Sistema operativo, consultado el 29 de septiembre del 2020
2. Wireshark, Software de análisis, consultado el 29 de septiembre del 2020
3. Docker, Software para contenedores, consultado el 29 de septiembre del 2020
4. Git, Software para control de versión, consultado el 29 de septiembre del 2020
5. Github, Sitio de repositorios git, consultado el 29 de septiembre del 2020
6. Oragono, Sitio del servidor utilizado, consultado el 29 de septiembre del 2020
7. Oragono, repositorio del servidor, consultado el 29 de septiembre del 2020
8. Weechat, Sitio del cliente utilizado, consultado el 29 de septiembre del 2020
9. Weechat, repositorio del cliente, consultado el 29 de septiembre del 2020
10. IRC Analysis, repositorio del trabajo
11. Documentación del protocolo, consultado el 29 de septiembre del 2020
12. Youtube, Vídeo del trabajo
13. Documentación del protocolo cliente (ingles), consultado el 29 de septiembre del 2020
14. Documentación del protocolo server (ingles), consultado el 29 de septiembre del 2020