Guión de pruebas TPE 2019/2

Última modificación: Sat, 19 Oct 2019 13:36:58 -0300

Esta es una versión preliminar de los casos de pruebas publicada con el objetivo de dar un acceso temprano. Es posible que en futuras revisiones se agreguen casos o se modifiquen los existentes (por ejemplo, faltan casos para line folding, y para verificar cantidad de conexiones concurrentes).

Revisiones

- Versión inicial.

Casos de prueba

0. Convenciones

`./pop3filter` se refiere al binario con el proxy POP3.

`./pop3ctl` se refiere al binario usado para obtener métricas de forma remota y obtener alguna configuración.

`./stripmime` se refiere al programa externo que implementa la censura de partes MIMEs.

Las trazas de TCP streams siguen las mismas convenciones que la funcionalidad *Follow TCP Stream* del Wireshark. Los bytes coloreados con azul son los que envía el cliente, mientras que los coloreados con rojo son los enviados por el servidor.
Cuando el caso de prueba pida verificar la salida, hay ciertos bytes donde se tolera alguna diferencia (por ejemplo el texto que continúa luego de -ERR ó de +OK). Por ejemplo el texto *listo para probar* en el siguiente ejemplo puede variar.

USER foo
+OK listo para probar

Se utilizará el estilo de comentarios de C para comentar alguna de líneas. Por ejemplo:

*/*Atención que LIST en este caso no es multilínea */*

En las transcripciones se utiliza el caracter … HORIZONTAL ELLIPSIS' (U+2026) para abreviar partes y facilitar la legibilidad del documento.

En todos los casos se ejecutará `pop3filter` y `stripmime` dentro de un entorno controlado de memoria. Para esto se puede utilizar el mecanismo provisto por `setrlimit(2)` mediante el comando `ulimit(1)`. En particular interesa el recurso `RLIMIT_AS` que limita la memoria virtual de un proceso.

Se puede ver en acción dicha limitación en el siguiente snippet:

```
$ ulimit -v $((1024*50))
$ cat 1.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(void) {
   for(unsigned i = 0; i < 1024 ; i++) {
       printf("%d\n", i);
       char * p = malloc(1024 * 1024);
       if(p == 0) {
```

```
            break;
        }
        memset(p, 0, 1024);
    }
    puts("ENTER para terminar");
    getchar();
}
$ gcc 1.c
$ ./a.out
0
…
45
ENTER para terminar
```

La intención es controlar la cantidad de memoria disponible para el programa independientemente de la memoria disponible del sistema. Cada implementador tendrá que determinar el número teniendo en cuenta su alocaciones.

En muchas de las pruebas se utiliza netcat, se ha utilizado las siguientes versiones:

```
$ nc --version
Ncat: Version 7.70 ( https://nmap.org/ncat )

$ nc -h 2>&1 |head -n5
OpenBSD netcat (Debian patchlevel 1.89-4ubuntu1)
This is nc from the netcat-openbsd package. An alternative nc is available
in the netcat-traditional package.
usage: nc [-46DdhklnrStUuvzC] [-i interval] [-P proxy_username] [-p
source_port]
        [-s source_ip_address] [-T ToS] [-w timeout] [-X proxy_protocol]
```

También se utiliza el comando `printf(1)` que es parte de coreutils GNU.

Se utiliza bash de interprete de comando. Por ejemplo

```
FILTER_MEDIAS=application/x-cd-image time cat
```

exporta la variable de entorno `FILTER_MEDIAS y luego` ejecuta `./time cat`
Si no se dispone de bash esto puede escribirse seguramente en dos comandos.

También se utiliza el comando `time(1)`. Es importante que el comando exista como comando y no como función built-in del shell.

```
[jcodagno@pampero ~]$ FILTER_MEDIAS=application/x-cd-image time cat
-bash:
time: command not found
[jcodagno@pampero ~]$ time
real  0m0.000s
user  0m0.000s
sys   0m0.000s
[jcodagno@pampero ~]$ which time
```

```
which: no time in
(/usr/local/sbin:/usr/local/bin:/usr/bin:/usr/lib/jvm/default/bin:/usr/
bin/site_perl:/usr/bin/vendor_perl:/usr/bin/core_perl)
```

Se puede modificar la forma en que se ejecuta el caso seteando previamente la variable (dejando de ser un one-liner) o se puede compilar y dejar en el PATH el comando time:

```
$ wget https://ftp.gnu.org/gnu/time/time-1.9.tar.gz
$ ./configure
$ make
$ ./time --version
time (GNU Time) 1.9
…
```

Se utiliza el comando `sha256(1)` para obtener un hash para detectar cambios inesperados. Puede ser reemplazado por alguna otro comando de hash similar (`md5sum(1)`, `sha1sum(1)`, `cksum(1)`) pero debe recalcularse los valores a comparar.

En muchos casos se hace referencias a nombres resolubles por DNS:

- *foo* host que corre el proxy POP3.
- *bar* host que corre el origin server.

Algunos casos utilizan un servidor pop3 simple implementado con AWK que debe ser expuesto a la red con un super server o con netcat o similar.

```
$ cat pop3.awk |head
#!/usr/bin/awk -f
BEGIN {
    printf "+OK hola!\r\n"
}
"QUIT" == toupper($1) {
   printf "+OK Adios!\r\n"
   exit 0
}
"USER" == toupper($1) && NF == 2 {
   printf "+OK Hola "$2"\r\n";
```

Versión para copiar, pegar y ejecutar:

```
$ cat pop3.awk | gzip -9 | base64
H4sIAPQ4q10CA5WSUU/CMBSF3/crLpMQiWHAHkd4mGZgI865jTdeKitSlNtl3ZSE8N9ZxaFQCKFv
Peee77a3vam1C5m13zi26fcHtGbGvTckPqwNKFeaccxnYN69PMFcfNLaJJugaWwM83VMYhP6fchF
kaYsu613m7vQ/4ybcCF/Q6XFVjyHjoqPIy/U4o0G+AMl2jrpsewOZt02f2A9ZSNb5T0FC9wougo2
zOiUUwkoQDIowdOCYS6AAtKEsxMtHtzAvXzdZy5hStMSntCESZACnT9YValYuqomoqsR8QMSeCPi
E3+o29aJo45IFJ+fRlc/dczwXUCBsGQo6YLtn6sq6oLd0cRrWtftM60PyHtM6MXhJczBL5tz+KIV
p3IGmVg6sCgoHjvHe/WzYVlgImrHlmWB07yUtyqhusF6b7W8cPeqsDG2FWoc7WsDAAA=
```

Una forma de exponerlo en la red en el puerto TCP 9001 es con el comando `socat`:

```
      ./socat TCP4-LISTEN:9001,crlf,reuseaddr
SYSTEM:'/tmp/pop3.awk',pty,echo=0
```

Si su sistema no dispone de socat siempre se puede compilar:

```
$ wget http://www.dest-unreach.org/socat/download/socat-1.7.3.2.tar.gz
$ tar xzf socat-1.7.3.2.tar.gz
$ cd socat-1.7.3.2/
$ ./configure
$ ./make -j2
```

1. Argumentos de las aplicaciones

Precondiciones:

- No se tiene corriendo ningún origin server en localhost.

100 - Binding en puertos default correcto

Ejecutar

```
Terminal A: $ ./pop3filter 127.0.0.1
Terminal B: $ netstat -nlp | grep pop3filter
  sctp        127.0.0.1:9090 LISTEN  13642/pop3filter
  sctp        ::1:9090       LISTEN  13642/pop3filter
  tcp         0.0.0.0:1110   LISTEN  13642/pop3filter
  tcp         :::1110        LISTEN  13642/pop3filter
```

Verificar:

- Puerto TCP 1110 escuchando en todas las interfaces.
- Puerto STCP 9090 escuchando únicamente en localhost.
- Ejecución exitosa de pop3ctl (obtención de métricas o configuración).

Es válido que también sctp aparezca sólo ::1 o 127.0.0.1 (o en algunos casos `:::1110` sin `0.0.0.0:1110`).

101 - Cambio de puertos e interfaces default

Ejecutar

```
Terminal A:
 $ ./pop3filter -p1111 -o 9091 -l ::1 -L0.0.0.0 127.0.0.1
Terminal B:
 $ netstat -nlp | grep pop3filter
  sctp 0.0.0.0:9091 LISTEN 13642/pop3filter
  tcp6 ::1:1111     LISTEN 13642/pop3filter
```

Verificar:

- Puerto TCP 1111 escuchando únicamente en localhost.
- Puerto STCP 9091 escuchando en todas las interfaces.
- Ejecución exitosa de pop3ctl (obtención de métricas o configuración).

2. Disponibilidad

200 - Origin server (IPV4) no presta servicio

Precondiciones:

- En 127.0.0.1 puerto 110 no se encuentra ningún servidor tcp

Ejecutar:

- `Terminal A: $ ./pop3filter 127.0.0.1`
- `Terminal B: $ nc -C 127.0.0.1 1110`

Verificar:

- En la terminal B debe aparecer un saludo negativo (`-ERR Connection refused`) y cerrar la conexión TCP.
  Alternativamente puede aparecer un saludo `+OK` y fallar ante cualquier otro comando. Se busca evitar aceptar la nueva conexión y cerrarla sin información al cliente.

201 - Origin server (IPV6) no presta servicio

Precondiciones:

- No se tiene corriendo ningún origin server en localhost.

Ejecutar:

- `Terminal A: $ ./pop3filter ::1`
- `Terminal B: $ nc -C 127.0.0.1 1110`

Verificar:

- En la terminal B debe aparecer un saludo negativo (`-ERR Connection refused`) y cerrar la conexión TCP.
  Alternativamente puede aparecer un saludo `+OK` y fallar ante cualquier otro comando. Se busca evitar aceptar la nueva conexión y cerrarla sin información al cliente.

202 - Origin server con múltiples direcciones IP (una falla)

Precondiciones:

- Origin server disponible en 127.0.0.1.
- El nombre `tpe.proto.leak.com.ar` resuelve a localhost y a una dirección IP que no está disponible

```
$ dig +short tpe.proto.leak.com.ar
240.0.0.1
127.0.0.1
```

Ejecutar:

- `Terminal A: $ ./pop3filter tpe.proto.leak.com.ar`

- `Terminal B: $ nc -C 127.0.0.1 1110`

Verificar:

- Saludo +OK, y se puede interactuar con el origin server por medio del proxy (USER, PASS, LIST).

203 - Comportamiento origin server resuelve DNS IPV6

Precondiciones:

- `bar.leak.com.ar.` es FQDN que resuelve una dirección IPV6 donde está está corriendo el origin server.
  ```
  $ dig +short -t AAAA bar.leak.com.ar
  ::1
  ```
- Terminal A: `$ ./pop3filter bar.leak.com.ar`

Ejecutar:

- Terminal B: `$ nc -C foo 1110`

Verificar:

- La presencia de un saludo positivo y que se puede interactuar con el servidor POP3.
- Se puede verificar adicionalmente con wireshark que los paquetes entre el proxy y el servidor origen se transfieren vía IPv6.

204 - Proxy Server como Origin Server (recursividad)

Precondiciones:

- `Terminal A: $ ./pop3filter -P 1110 localhost`

Pasos a seguir:

```
Terminal B: $ nc -C foo 1110
```

Verificar

- Nota: El comportamiento depende de la implementación.
- Una implementación consciente del problema detecta la recursión antes de conectarse y respondería con un mensaje de -ERR.
- Una implementación no consciente (es lo tradicional), comienza a conectarse en cascada hasta que se utilizan todos los file descriptors. En tal caso, la última conexión que falla comienza a liberar los recursos nuevamente, y el proxy se tiene que poder que seguir utilizando normalmente.

205 - Performance e inmutabilidad de bytes

Precondiciones:

- Origin server disponible en el host `bar`.
- Usuario con al menos un correo electrónico (lo llamaremos `foo con contraseña foo`).
- `Terminal A: ./pop3filter bar`

Se trabajará con el I - Mensaje “Grande” para tener mejores mediciones.

En primer lugar se calcula primero la línea base yendo directo contra el servidor pop3:

```
$ time printf 'USER foo\nPASS foo\nRETR 1\n'|nc -C bar 110|pv|tail -n +5|head
-n -1|sha256sum
 662MiB 0:00:03 [ 194MiB/s] [   <=>                                  ]
5e95fc3477fba33b6f9574bad93bbe4c4a55593f049a3cde644546fceaad3916  -

real   0m3.413s
user   0m3.455s
sys    0m1.663s
```

Para luego calcular el tiempo utilizando el proxy:

```
$ time printf 'USER foo\nPASS foo\nRETR 1\n'|nc -C foo 1110|pv|tail -n
+5|head -n -1|sha256sum
```

Nótese que no importa comparar los números de forma absoluta (por ejemplo el primer paso podría tardar 6 segundos), si no que interesa la relación entre las dos ejecuciones (es decir depende de cada computadora).

Post-condiciones:

- El sha256 debe coincidir en ambas ejecuciones (y a su vez coincidir con I - Mensaje “Grande”)
- El tiempo de transferencia utilizando el proxy tiene que estar dentro del mismo orden que la la transferencia utilizando el proxy.

206 - Concurrencia

Precondiciones:

- Mismas que el caso 205.

Se realiza los mismos pasos que en el caso 205 pero en simultáneo en 4 terminales.

Nótese que no importa comparar los números de forma absoluta (por ejemplo el primer paso podría tardar 6 segundos), si no que interesa la relación entre las dos ejecuciones.

Nótese que sha256 consume bastantes ciclos de cpu. En caso de tener una máquina reducida lo puede cambiar por un comando más liviano.

Verificar:

- El sha256 debe coincidir en todas las ejecuciones (y a su vez coincidir con el calculado al inicio de este documento)
- El tiempo de transferencia utilizando el proxy tiene que estar dentro del mismo orden que la la transferencia utilizando el proxy. El tiempo de transferencia puede incrementarse en 4x.

207 - Desconexión repentina de clientes

Precondiciones:

- Origin server disponible en el host `bar`.
- Usuario con al menos un correo electrónico (lo llamaremos `foo con contraseña foo`).
- `Terminal A: ./pop3filter bar`

En Terminal B:

```
$ nc -C foo 1110
+OK listo para probar
```

En Terminal C

- `kill -9 <pid de nc de Terminal B>`

Verificar:

- que netcat haya terminado
- que el proxy siga ejecutando y recibiendo nuevas conexiones y no se esté consumiendo todo el CPU.

208 - Desconexión repentina del origin server esperando comando

Precondiciones:

- Origin server disponible en el host `bar`.
- Usuario con al menos un correo electrónico (lo llamaremos `foo con contraseña foo`).
- `Terminal A: ./pop3filter bar`

En Terminal B:

```
$ nc -C foo 1110
+OK listo para probar
```

En Terminal C

- terminar el servidor pop.

Verificar:

- que el servidor POP3 no esté disponible.
- que el proxy siga ejecutando y recibiendo nuevas conexiones y no se esté consumiendo la mayoría del CPU.

- se haya terminado la conexión de la Terminal B.
  Alternativa I: si no terminó por lo menos que termine la conexión al enviar el siguiente comando.

209 - Desconexión repentina del origin server durante un RETR

Precondiciones:

- Origin server disponible en el host `bar`.
- Usuario con al menos un correo electrónico (lo llamaremos `foo con contraseña foo`).
- `Terminal A: ./pop3filter bar`
- Se obtendrá el mensaje más grande para tener tiempo de matar el servidor pop origen: I - Mensaje “Grande”.

En Terminal B:

```
$ nc -C foo 1110
+OK listo para probar
USER foo
+OK
PASS foo
+OK
RETR 1
```

En Terminal C

- terminar el servidor POP3.

Verificar:

- que el servidor POP3 no esté disponible.
- que el proxy siga ejecutando y recibiendo nuevas conexiones y no se esté consumiendo la mayoría del CPU.
- se haya terminado la conexión de la Terminal B.
  Alternativa I: si no terminó por lo menos que termine la conexión al enviar el siguiente comando.

210 - Lecturas parciales desde el cliente

Pre-condiciones:

- Todas las transformaciones y opciones apagadas
- origin server `pop3.awk` (aunque puede ser cualquier otro). Lo importante es enviar un caracter por paquete IP.

Pasos a seguir.
Se trabajará en POP3 enviando comandos cada tecla con una separación de medio segundo, logrando que cada byte se envíe en su propio paquete IP.

El vídeo netcat-enviar-un-byte-por-vez.webm distribuido con este material muestra cómo hacerlo en Linux (para HTTP y no para POP3, pero semánticamente no hay diferencia):

```
$ stty -icanon &&  nc -C  127.0.0.1 1110
+OK hola!
USER foo
+OK Hola foo
PASS foo
+OK Gracias no se la cuento a nadie
RETR 1
+OK Ahi va
From: juan

hola mundo!
.. :)

.
QUIT
+OK Adios!
```

Post-condiciones:

- Debemos observar la respuesta de forma correcta.

3. Uso del proxy POP3

301 - Comportamiento origin server que da servicio comandos básicos

Precondiciones:

- Origin server disponible en el host `bar`.
- Usuario con al menos un correo electrónico (lo llamaremos `foo con contraseña foo`).
- `Terminal A: ./pop3filter bar`

El objetivo es verificar un abanico amplio de comandos POP3.

`Terminal B: nc -C foo 1110` donde se realiza la siguiente interacción:

```
+OK listo para probar
CAPA
+OK
CAPA
TOP
UIDL
RESP-CODES
PIPELINING
.
CAPA 123 /* El argumento no está en el RFC, pero bien puede ser transparente.  Puede dar -ERR
+OK          si la implementación no copia transparentemente o el origin server es estricto
CAPA       */
TOP
UIDL
RESP-CODES
PIPELINING
.
USER foo
+OK
PASS foo
+OK
NOOP
+OK
STAT
+OK 2 1266
LIST
+OK 2 messages:
1 630
…
.
LIST 1              /*Atención que  LIST en este caso no es multilínea */
+OK 1 630
```

```
LIST 5000          /*Atención que los LIST que fallan no son multilínea */
-ERR There's no message 5000.
DELE 1
+OK Marked to be deleted.
RSET 0             /*muy importante para no tener que rearmar el caso*/
+OK
RETR 1
+OK 630 octets
…
.
RSET
+OK
TOP 1 1
+OK
Return-Path…
.
UIDL
+OK
1 0000000457cf5d98
2 0000000557cf5d98
.
UIDL 1
+OK 1 0000000457cf5d98
```

302 - Pipelining cliente

Precondiciones:

- Mismas que el caso 301 .

El objetivo es verificar un abanico amplio de comandos POP3 utilizando pipelining en el cliente.

En Terminal B:

```
$ printf 'USER foo\nPASS foo\nLIST\nTOP 1 1\nQUIT\n'  | nc -C foo 1110
+OK listo para probar
+OK
+OK Logged in.
+OK 2 messages:
1 640
2 704
.
+OK
From: …
.
+OK Logging out.
```

303 - Trailing spaces no afectan a los comandos

Similar a 302 - Pipelining Cliente pero agregando espacios al final de un comando.
Se espera la misma salida que dicho caso.

```
$ printf 'USER foo\nPASS foo\nLIST  \nTOP  1   1 \nQUIT      \n'  |nc -C foo 1110
+OK listo para probar
+OK
+OK Logged in.
+OK 2 messages:
1 640
2 704
.
+OK
From: …
.
+OK Logging out.
```

4. Pipelining

401 - Pipelining con origin server que no soporta pipelining (de forma declarada)

Precondiciones:

- Servidor origen ejecutando `pop3.awk`.
- Terminal A: `$ ./pop3filter bar`

Ejecutar:

- `printf "CAPA\nUSER foo\nPASS foo\nLIST\n"| nc -C foo 1110`

Verificar:

- La salida es:

  ```
  +OK hola!
  +OK Mis capacidades son:
  CAPA
  USER
  SINPIPELINING
  PIPELINIG
  .
  +OK Hola foo
  +OK Gracias no se la cuento a nadie
  +OK Tengo un mensaje
  1 20
  .
  ```

  La incorporación de PIPELINING puede ocurrir en cualquier posición de la lista. El servidor de forma adrede envía la capacidad `SINPIPELINING` para verificar que se esté interpretando correctamente las capacidades (y que no se haga un búsqueda sin tener principio y fin)
- Como el origin server no soporta pipelining se ha enviado un comando por vez.
  `sudo tcpdump -s0 -Al 'tcp port 110'`

402 - Pipelining hacia origin server (declara que lo soporta)

Precondiciones:

- Servidor origen ejecutando soporta CAPA y PIPELINING
- Terminal A: `$ ./pop3filter bar`
- Terminal B: `sudo tcpdump -s0 -Al 'tcp port 110'`

Ejecutar:

- `printf "USER foo\nPASS foo\nLIST\n"| nc -C foo 1110`

Verificar:

- En la terminal B se tiene que observar que los mensajes se enviaron en bulk hacia el origin server.

5. Transformaciones de mensajes (standalone)

Todos los casos correrán con las siguientes variables de entornos. Algunos casos pisarán alguna de estas variables.

```
export FILTER_MEDIAS="images/png"
export FILTER_MSG="Parte reemplazada."
export POP3FILTER_VERSION=0.0
export POP3_USERNAME=foo
export POP3_SERVER=bar
```

Advertencia: Puede admitirse cambios que mantengan la semántica (orden de headers, case de nombres de headers), etc. Es por esto que se recomienda utilizar diff para revisar las salidas en vez de un hash/message digest.

501 - Inmutabilidad

Si el correo no tiene la parte que se está buscando no debería cambiar.

Ejecutar:

```
  FILTER_MEDIAS=application/json ./stripmime < ii_images.mbox >
out
```

Verificar:

- que no existan cambios.
  ```
  diff -u ii_images.mbox out
  ```

502 - Standalone: Simple PNG

Ejecutar:

```
FILTER_MEDIAS=image/png ./stripmime < ii_images.mbox > out
```

Verificar:

- que únicamente se censure la parte png
  ```
  diff -u ii_images.mbox out
  ```

503 - Standalone: PNG y JPEG: lista

Ejecutar:

```
FILTER_MEDIAS=image/png,image/jpeg ./stripmime <
ii_images.mbox > out
```

Verificar:

- que únicamente se censure las partes jpeg y png
  ```
  diff -u ii_images.mbox out
  ```

504 - Standalone: PNG y JPEG: wildcard

Ejecutar:

```
FILTER_MEDIAS=image/* ./stripmime < ii_images.mbox > out
```

Verificar:

- que únicamente se censure las partes jpeg y png
  `diff -u ii_images.mbox out`

505 - Anidado PNG y JPEG: wildcard

Ejecutar:

```
FILTER_MEDIAS=image/* ./stripmime < iii_images_fwd.mbox > out
```

Verificar:

- que únicamente se censure las partes jpeg y png
  `diff -u ii_images.mbox out`

506 - Correo grande: Inmutabilidad

Se busca un tipo de parte que no está presente en correo, el correo es grande, no tendría que modificarse.

Ejecutar:

- `time cat i_big.mbox  > /dev/null`
  `FILTER_MEDIAS=a/b time cat i_big.mbox | ./stripmime | pv > out`

No debe haber diferencias en la salida (idealmente el checksum debe coincidir con III - Forward de II).

También es importante ver la relaciones de tiempo de ejecución entre cat (sin filtro) y stripmime (sin filtro).

Se puede tomar como línea base para comparar performace

- `FILTER_MEDIAS=a/b time cat ../mensajes/i_big.mbox | awk '{if(NF>=0)print $0}' | pv > out`

507 - Correo grande: Mutabilidad

Se busca la parte de más megas del mensaje.

Ejecutar:

```
FILTER_MEDIAS=application/x-cd-image time cat i_big.mbox | ./stripmime | pv > out
```

Verificar:

- que únicamente la parte application/x-cd-image quede fuera del mensaje.
- es interesante también comparar los tiempos de ejecución

6. Integración de la Transformaciones de mensajes

601 - Transformación utilizando cat

Precondiciones:

- Origin server disponible en el host `bar`.
- Usuario con al menos un correo electrónico (lo llamaremos `foo con contraseña foo`). Allí están los mensajes de prueba.
- `Terminal A: ./pop3filter -t 'touch /tmp/1 && cat' bar`

La intención del caso es corroborar que se ejecuta el proceso externo para transformar. Utilizamos cat que no realiza ninguna modificación sobre el mensaje.

En la terminal B asegurarse de que no exista el archivo /tmp/1 (rm -rf /tmp/1), y se obtienen los mensajes (ej: `$ printf 'USER foo\nPASS foo\nRETR 1\nQUIT'|nc -C 127.0.0.1 1110 > out`)

Se debe verificar que:

- se cree el archivo `/tmp/1`.
- no existan diferencias en el mensaje (diff out msg)

602 - Seteo de variables de entorno

Precondiciones:

- Origin server disponible en el host `bar`.
- Usuario con al menos un correo electrónico (lo llamaremos `foo con contraseña foo`). Allí están los mensajes de prueba.
- `Terminal A: ./pop3filter -Mimage/jpeg -m holamundo -t 'bash envs.sh' bar`
  Donde envs.sh:

```
#!/bin/bash
cat << EOF | unix2dos
X-Media: $FILTER_MEDIAS
X-Version: $POP3FILTER_VERSION
X-User: $POP3_USERNAME
X-Origin: $POP3_SERVER

Body.
$FILTER_MSG
```

```
EOF
cat > /dev/null
```

Se debe obtener un mensaje usando el proxy e independientemente de el mensaje origen debe obtenerse:

RETR 1
+OK
X-Media: image/jpeg
X-Version: 0.0.0
X-User: juan
X-Origin: bar

Body.
holamundo

.

603 - Respuesta lazy

Precondiciones:

- Origin server disponible en el host `bar`.
- Usuario con al menos un correo electrónico (lo llamaremos `foo con contraseña foo`). Allí están los mensajes de prueba.
- `Terminal A: ./pop3filter -t 'bash lazy.sh'  bar`
  Donde `lazy.sh`:

```
#!/bin/bash
cat > /tmp/foo
cat /tmp/foo
```

En este caso se verifica que el proxy no espere un flujo de datos específico. El proceso transformador consume todos los bytes para recién luego emitirlos. Se debe verificar en el cliente que el mensaje no contenga diferencias. Se puede probar con un mensaje pequeño primero, y luego con el mensaje grande.

604 - Byte Stuffed

Precondiciones:

- Origin server disponible en el host `bar`.
- Usuario con al menos un correo electrónico (lo llamaremos `foo con contraseña foo`). En este caso no importa el contenido ya que será reemplazado.
- `Terminal A: ./pop3filter -t bytestuffed.sh bar`
  Donde `bytestuffed.sh`:

```
#!/bin/bash
cat > /dev/null
cat << EOF | unix2dos
X-Header: true
```

```
.hola
.
EOF
```

Al obtener el mensaje tanto “.hola” como “.” deben estar escapados:

```
$ nc -C localhost 1110
+OK listo para probar
USER juan
+OK
PASS juan
+OK Logged in.
RETR 1
+OK
X-Header: true

..hoola
..
.
```

Mensajes

Algunos mensajes son autogenerados en base a ciertos comandos. Esta guía ha sido probado con las siguientes versiones. Las versiones son de una distribución Linux “vieja” (2011) y de una distribución nueva (2017) como para garantizar cierta *portabilidad*.

Las diferentes partes que hace referencia en esta sección se encuentran en el archivo *mensajes.tar.gz.*

```
$ dos2unix  --version
dos2unix 5.3.1 (2011-08-09)
With native language support.
LOCALEDIR: /usr/share/locale

$ dos2unix --version
dos2unix 7.4.0 (2017-10-10)
With Unicode UTF-16 support.
With native language support.
LOCALEDIR: /usr/share/locale
http://waterlan.home.xs4all.nl/dos2unix.html

$ base64 --version
base64 (GNU coreutils) 8.13
Copyright (C) 2011 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Written by Simon Josefsson.

$ base64 --version
base64 (GNU coreutils) 8.29
Copyright (C) 2017 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Written by Simon Josefsson.
```

I - Mensaje “Grande”

Generamos un archivo grande para realizar pruebas mediante una secuencia de pasos.

0. Obtener el paquete de correos de prueba, y hacerlo el current directory.

1. Descargar archivo de prueba.

```
$ wget
https://archives.fedoraproject.org/pub/archive/fedora/linux/releases/26/Server/x86_64/iso/Fedora-Server-netinst-x86_64-26-1.5.iso
```

2. Verificar descarga exitosa. La salida del siguiente comando debe coincidir.

```
$ sha256sum Fedora-Server-netinst-x86_64-26-1.5.iso
e260921ef5c7bd5ee2a7b2f2f1156af6483014c73984e4cf37f2b6690e0155e5
Fedora-Server-netinst-x86_64-26-1.5.iso
```

3. Verificar que el comando base64 genere líneas de 76 caracteres. De no ser así modificar los comandos que aparecen en la guía para que se genere la misma salida y de esta forma sea sencilla la comparación. Típicamente la opción -w controla el ancho de cada línea.

```
$ base64 < Fedora-Server-netinst-x86_64-26-1.5.iso |head -n2
RVIIAAAAkJAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAz7fqO1bwAfPv8ZjHbZjHJZlNmUQZXjt2O
xVK+AHy/AAa5AAHzpepLBgAAUrRBu6pVMckw9vnNE3IWgftVqnUQg+EBdAtmxwbxBrRC6xXrAFpR
```

4. Armado del mensaje.

```
$ cat i_big.head > i_big.mbox
$ base64 < Fedora-Server-netinst-x86_64-26-1.5.iso | unix2dos >>
i_big.mbox
$ cat i_big.tail >> i_big.mbox
$ sha256sum i_big.mbox
5e95fc3477fba33b6f9574bad93bbe4c4a55593f049a3cde644546fceaad3916  i_big.mbox
$ ls -l i_big.mbox
-rw-rw-r-- 1 juan juan 694489434 Nov  5 12:57 00_big.mbox
$ wc -l i_big.mbox
8903731 i_big.mbox
```

Nótese que tomamos las precaución de generar los archivos con líneas que terminan con CR LF para que sea más sencillo de comparar. Esto asume que servidor POP3 origen no modificará el mensaje.
Debe verificarse esa hipótesis obteniéndose el mensaje y comparando el checksum:

```
$ printf 'USER foo\nPASS foo\nRETR 1\n'|nc -C bar 110|tail -n +5|head -n -1 |sha256sum
5e95fc3477fba33b6f9574bad93bbe4c4a55593f049a3cde644546fceaad3916  -
```

II - Multipart imágenes y txt

El archivo `ii_images.mbox` contiene un `multipart/mixed` que a su vez contiene una parte `text/plain`, una parte `image/png`, una parte `image/jpeg`, y una parte `text/plain`.

III - Forward de II

El archivo `iii_images_fwd.mbox` contiene una parte `multipart/mixed` que a su vez contiene una parte `text/plain` y otra parta `message/rfc822` con el mensaje II - Multipart imágenes y txt.