SCHOOL OF HACKING 2015
RETO BUFFER OVERFLOW

INSTRUCCIONES

Para la resolución del reto habrá que tener una máquina física o virtual Linux instalada al menos con las siguientes características:

- Compilador gcc. La aplicación en el sistema vulnerable se ha compilado con la versión gcc 4.4.3
- netcat
- perl

El fichero `reto.c` con el código fuente de la aplicación vulnerable se debe copiar a una máquina LINUX para su examen y pruebas. Para la resolución del reto se seguirán los pasos que se sugieren a continuación.

La aplicación vulnerable estará disponible para ser atacada solamente durante un periodo de tiempo que se anunciará por email a los que se han inscrito en el reto. Esta aplicación estará disponible en los puertos TCP 5000 a 5015 de la máquina víctima. También en este email se indicará la IP del servidor. En caso de que la aplicación deje de funcionar en un puerto debido a la interacción del ataque, hay que tener en cuenta que se puede acceder en alguno de los otros puertos disponibles. El puerto atacado volverá a estar disponible en un plazo de un minuto.

Está ABSOLUTAMENTE PROHIBIDO realizar escaneos, pruebas y ataques contra la máquina servidor que no sean los que se contemplan en los ejercicios a realizar. Se registrarán los accesos para poder comprobar que no se saltan estas reglas.

Solamente hay tres ejercicios que puntuarán para la consecución del reto. Cuando el participante crea haber resuelto estos ejercicios deberá mandar un email a la dirección gmacia@ugr.es conteniendo de forma breve y clara la solución a los 3 ejercicios, con la plantilla de la siguiente página. Solamente se aceptará una respuesta por participante. Cada ejercicio puntuable será evaluado por separado. La puntuación se obtendrá por el orden en el que se da la solución.

Ejemplo. Los ejercicios A, B y C valen, respectivamente 10, 15 y 20 puntos. Pepe manda la primera respuesta correcta a ejercicios A y B, pero errónea (o incompleta) para el C. Después de Pepe, Juan manda la solución correcta a los ejercicios A, B y C.
Puntuación de Pepe: 10 puntos (A) + 15 puntos (B), 0 puntos (C) = 25 puntos
Puntuación de Juan: 9 puntos (A), 14 puntos (B), 20 puntos (C) = 43 puntos

EJERCICIO 8 (10 puntos):

Nombre de la función: _____

Número de la línea: _____

Código en dicha línea: _____

EJERCICIO 9 (15 puntos):

Dirección IP desde la que se ha hecho el ejercicio: _____

Fecha y hora en la que se ha realizado con éxito: _____

Cadena introducida: _____

EJERCICIO 10 (20 puntos):

Dirección IP desde la que se ha hecho el ejercicio: _____

Fecha y hora en la que se ha realizado con éxito: _____

Password obtenida: _____

EJERCICIOS A REALIZAR

Los ejercicios a realizar son los siguientes:

1. Examina el código fuente del programa y trata de averiguar qué hace.

2. Compila el programa con el siguiente comando (desde el directorio donde está reto.c):

```
gcc -g -fno-stack-protector -z execstack -o reto reto.c
```

3. Deshabilita la protección de aleatorización (ASLR):

```
sudo su
echo 0 > /proc/sys/kernel/randomize_va_space
```

4. En una terminal ejecuta el programa para que escuche en un puerto (ej. 5000):

```
./reto 5000
```

5. Conéctate desde otro terminal a la aplicación vulnerable desde la misma máquina:

```
nc  <nombre_servidor>  5000
```

   donde `nombre_servidor` será `localhost` si la aplicación se ejecuta en nuestra máquina.

6. Haz pruebas introduciendo passwords diferentes y trata de adivinar con el código del programa la contraseña que te da acceso al sistema.

7. Prueba con la aplicación corriendo en el servidor oficial del reto UCyS.

¿Funciona dicha clave en el servidor también?

8. **(PUNTUACIÓN: 10 Puntos)** Identifica la vulnerabilidad de buffer overflow que se podría explotar para tener acceso al sistema sin conocer la password. ¿En qué función aparece la vulnerabilidad? ¿En qué línea del fichero `reto.c` aparece el código vulnerable (enviar en la respuesta el número de línea y el código de dicha línea)? NOTA: Tomar la línea del fichero original, no del modificado por el participante.

9. **(PUNTUACIÓN:15 puntos)** Adivina qué cadena podríamos introducir como password para entrar en el sistema sin conocer la clave. Prueba en tu máquina local y también en el servidor oficial del reto UCyS.
   Pista: estudia bien la función check_password.

10. **(PUNTUACIÓN: 20 puntos)** Trata de explotar una vulnerabilidad de la aplicación para obtener la password almacenada en el servidor oficial del reto UCyS.
    Pista: trata de ejecutar la función show_password. (4 puntos).

Pista general: para mandar una cadena de texto creada con perl ejecutar:

```
perl -e 'print "cadena"' | nc <servidor> <puerto>
```

CÓDIGO DE LA APLICACIÓN VULNERABLE

```
/*
 *     APLICACIÓN reto.c
 *     RETO UCyS - 2015
 *     By: Gabriel Maciá
 */

#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <time.h>

char password_global[30] = "";
int socket_global;

int show_password () {

       int sock = socket_global;
       write (sock, "The password is: ", 17);
       write (sock, password_global, strlen(password_global));
       write (sock, "\n", 1);

       return 0;
}


int check_password (char *pass) {

       char password[32] = "hacking";
       char buf[32];
       memset (buf, '\0', 32);
       if (strcmp (password_global, "") != 0) {
              strcpy (password, password_global);
       }
       strcpy (buf, pass);

       if (strncmp (buf+2, password+2,4)==0 && strncmp(buf, "ha",2)==0) {
              return 1;
       } else {
              return 0;
       }

}

void imprime_banner(int sock) {

       char mensaje[500];

       FILE *f = fopen ("/home/admin/BufferOverflowSeminar/Reto/banner.txt",
"rt");
       if (f != NULL) {
              while (fgets(mensaje, 500, f) != NULL) {
                     write (sock, mensaje, strlen(mensaje));
              }
       }
       strcpy (mensaje,"------------------------------------------------------
-------\n    BIENVENIDO AL SISTEMA RETOS UCyS\n------------------------------
------------------------------\n      Introduzca la contraseña: ");
       write (sock, mensaje, strlen(mensaje));
```

```
}



int main(int argc, char *argv[])
{
    int sockd = 0, connfd = 0;
    struct sockaddr_in serv_addr;
    struct sockaddr_in cliente;
    size_t addr_len = sizeof(struct sockaddr_in);
    char ip_cliente [INET_ADDRSTRLEN];
    char sendBuff[1025];
    char recvBuff[1025];
    time_t ticks;

    if (argc < 2) {printf ("Usage: ./reto <#puerto>\n"); exit(0); }

       // Lectura de password del fichero
       FILE *fp = fopen ("/home/admin/BufferOverflowSeminar/Reto/password.txt",
"r");
    if (fp != NULL) {
             fscanf (fp, "%s", password_global);
             fclose(fp);
    }


       // Apertura de socket para escuchar conexiones entrantes
    if ((sockd = socket(PF_INET, SOCK_STREAM, 0)) == -1){
             perror("Error al crear socket\n");
             close(sockd);
             exit (0);
       }
    memset(&serv_addr, '0', sizeof(struct sockaddr_in));
    memset(sendBuff, '\0', 1025);
    memset(recvBuff, '\0', 1025);

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(atoi(argv[1]));

    if (bind(sockd, (struct sockaddr*)&serv_addr, sizeof(serv_addr))<0) {
                   perror ("Error al hacer bind\n");
                   exit(0);
       }

    if (listen(sockd, 1) < 0) {
                   perror ("Error en listen\n");
                   close(sockd);
                   exit(0);
       }

       // Bucle principal
    while(1)
    {

             // Aceptación de nuevas conexiones por parte de clientes.
             connfd = accept(sockd, (struct sockaddr *)&cliente, &addr_len);
        if (connfd == -1) {
                        perror ("Error en accept\n");
                        close (sockd);
                        close (connfd);
                        exit(0);
             }
             socket_global = connfd;

             // Envío de mensaje de pregunta de contraseña y lectura de
contraseña
             imprime_banner (connfd);
```

```
            int n = read (connfd, recvBuff, sizeof (recvBuff)-1);
            recvBuff[n-1] = '\0';

            //Conversion de la IP del cliente para imprimirla e impresion de
la contraseña recibida y hora de recepcion
            inet_ntop(AF_INET, &cliente.sin_addr.s_addr, ip_cliente,
INET_ADDRSTRLEN);
            ticks = time(NULL);
            snprintf(sendBuff, sizeof(sendBuff), "%.24s: ", ctime(&ticks));

            printf ("%s  %s     %s     ", sendBuff, ip_cliente, recvBuff);
            fflush(stdout);


            // Comprobacion de la contraseña recibida
            int resultado = check_password (recvBuff);
            if (resultado) {
                  printf ("ACCESO PERMITIDO\n");
                  fflush(stdout);
                  strcat (sendBuff, "\n\n   -----> ACCESO PERMITIDO
!!!!!!!!!!!!\n\n");
            } else {
                  printf ("password incorrecta\n");
                  fflush(stdout);
                  strcat (sendBuff, "\n\n          Password incorrecta\n\n");
            }

            //Se envia el resultado al cliente
            write(connfd, sendBuff, strlen(sendBuff));
            close(connfd);
      }
}
```