Mariona Farré

Sistemes Operatius
Lab 2 . Preparant als replicants

Enigma 1
(LAB 2) BronceKey
La resistència ha interceptat un algoritme que el Capitol utilitza per autenticar als agents que volen accedir al banc de temps. La clau per a saber el seu funcionament està en les vegades que s'executa la funció CheckSum().

```
void CheckSum (){ ...}
void main (int argc, char *argv[])
{ int i; int pid=0;
for (i = 0; i < 9; i++) {
if (pid == 0){
     pid = fork ();
     CheckSum(); }
}
while ((waitpid(-1, NULL, 0) > 0));

}
```

Calcula el número de vegades que s'executa la funció Checksum().
El valor obtingut s'haurà d'expressar com a un enter negatiu de 16 bits en Ca2 representat en hexadecimal.
Exemple: si el valor obtingut es 7, el valor de l'enigma serà FFF9

Objectius d'aquesta sessió

- Creació de programes a partir de diferents arxius. Ús de llibreries
- Comprensió de la taula de canals. Redireccionament de canals d'entrada/sortida

La resistència ens ha proporcionat un «generador de Banc de Temps» anomenat gendummy [S/L], que ens permet crear un banc de temps fictici per tal d'entrenar als replicants. Una vegada activat el generador, es crea un Banc de Temps anomenant dummy.dat de mida:

- 200MB de temps si es crida amb argument S (small)
- 1GB de temps si es crida amb argument L (large)

A mes, també ens proporcionen un altre dummy, el dummy2 que substituirà a l'anterior.
llibreria dummy substituida per dummy2
Aquest conté un conjunt de funcions que ens permetran fer les següents accions:

- dummy_open: obra el banc de temps, comprova que sigui un banc de temps correcta i ens retorna un canal per a poder **accedir**. en dummy.dat (escriure,llegir)
- dummy_comp (buffer, bytesllegits): cada vegada que un replicant roba un bloc de N unitats de temps, l'ha de processar en aquesta funció per a convertir-ho ens temps de vida real. (buffer guardar el temps robat, bytes a robar)
- dummy_test(acum, ENIGMA1, TEAMNAME): ens diu si, una vegada finalitzat l'atac i processat tot el temps obtingut, el processament s'ha realitzat correctament. (acumulat,enigma a trobar,nompropi) retorna si s'ha realitzat correctament (tot el temps)

- dummy_init(buffer, N): prepara el contenidor de memòria del replicant (prepara array de chars i cada bloc)
- dummy_exit(): permet que el replicant surti del Capitol sense deixar cap empremta. A més envia el seu codi de finalització al TT per a que el pugui recuperar. (envia a TT, els acomula i verificar que s'ha robat tot el temps)

1. Time Thief TT

Una vegada generat el banc de temps, el TT ho ha d'obrir, després d'haver comprovat que els valors d'entrada son correctes, i abans de crear els replicants.
A continuació, el TT ha de crear els replicants i esperar a que acabin d'atacar al nostre banc de temps.
A mida que els replicants vagin finalitzant la seva missió, el TT ha de recuperar el codi de finalització de cadascú (valor entre 0 i 127) i l'acumularà amb la resta de codis dels altres replicants. Aquest valor serà entregat a la funció dummy2_test(acum) (tots els codis de finalitzacio)per saber si el processament s'ha realitzat correctament (si es tot el banc del temps). En cas que qualsevol dels replicants no hagi pogut completar la seva missió amb èxit, el TT avortarà la missió d'atac i enviarà un missatge informatiu.

a) Identifica l'enunciat amb els continguts de l'assignatura

coses en blau (mig resum)

b) Realitza un pseudocodi que expliqui clarament, en termes de l'assignatura, com es prepara el TT per l'atac al banc de temps.

Pseudocodi i on estan les diff coses

c) Modifica el TT amb les noves accions. El codi ha de contenir:

#define TEAMNAME "" // insert between "" your team name
#define ENIGMA1 ""// insert between ""solution of enigma1

Constants definides #define TEAMNAME "mariona.farre" (mateix per tots els laboratoris)
#define ENIGMA1 ""

El TT obre el fitxer del banc de temps amb dummy_open() → retorna un canal per poder accedir al fitxer (podem accedir a dummy.dat→llegir les dades extreure dades)
TT crea replicants→ fork (creant replicants) mutar amb exelp…altres i esperar waitpid/wait
Aquests replicants acaben d'atacar→ passar el codi d'acabar i anar sumant amb si mateixos (acomularlos tots)
Passar l'acumulador a dummy2_test(acum)
SI un replicant == error para i envia missatge informatiu

2. Replicant

Prepara al replicant per a que una vegada hagi comprovat que el valor N, mida del bloc de temps que ha de robar, es correcte, realitzi les següents accions:

- Prepari el seu contenidor de memòria funcio (dummy_init())
- Comenci a obtenir blocs de N unitats de temps del banc de temps fictici que ha obert el TT.

(llegir-read(__, ___, ___)- les dades de bloc N entri en l'array de temps que es pot guardar)
guardar els bytes de N en l'array del replicant

- Converteixi el bloc de temps robat en temps de vida real.

funcio (dummy_comp(__,__)) converteix en vida real

- surti del Capitol sense deixar cap empremta i envii el seu codi de finalització al TT per a que el pugui recuperar.

funcio (dummy_exit ()) esperar que acabin els replicants i enviar a TT

a) Identifica l'enunciat amb els continguts de l'assignatura
b) Realitza un pseudocodi que expliqui clarament, en termes de l'assignatura, com es prepara el Replicant per l'atac al banc de temps.
c) Modifica el Replicant amb les noves accions.

Crear TT→ Va creant replicants R1…. Rn
Esperar que acabin
replicants ROBAR el temps del banc→ accedir a dummy.dat()
Processar el temps i sumar en acumulador
Al acabar de processar tot el temps enviar-ho a TT

Taula de Dispositius Virtuals Oberts:TDVO:es crea la taula al crear replicants cada dispositiu
tres canals (0=standar input,1=standar output,2=standar error)
= teclat =pantalla =pantalla
Sistema crea TFO: surt totes les des des de tots els dispositius oberts (tots els fitxers)

File descriptors: ex: si es vol llegir fer: read(0, &c, 1) (entrada,on guardar,tamany en bytes)
escriure els resultats en pantalla: write(1, &c, 1) (sortida,que treure per pantalla,en bytes)

Llegir de un file descriptor-del dummy.dat
fd=open( ) un dispositiu en TDVO que seria el fitxer dummy.dat
read (fd,___,___)

close (0) //tancar canal de input el teclat
close (fd) //tancar el fitxer en TDVO en dummy.dat

Pq els replicants puguin agafar datos de dummy.dat

replicant abans de mutar→ ha de llegir del fitxer
El TT vol llegir valors del fitxer no del teclat com a standar input→ redireccionament d'entrada
EX:

```
fd=open()     //nou dispositiu creat que sera el fitxer de dummy.dat
close (0)     //tancar entrada de teclat i queda lliure
 dup(fd)      //busca primer canal buit (que seria el canal 0-entrada)i passa el fitxer
              de  document dummy.dat
close (fd)    //tancar fd -ja no necessari
```

Replicant ha de llegir el fitxer:

```
read (0,__,__) //0=ara correspon al fitxer dummy.dat
```

generar el dummy.dat que estigui en el mateix directori (fitxer)

tt

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include "dummy2.h"

#define TEAMNAME "mariona.farre"
#define ENIGMA1 "FFEE"

void main(int argc,char *argv[]){
	int M, N, pid, i, exitcode, status, fd;
	M=atoi(argv[1]);
	N=atoi(argv[2]);
	fd = dummy_open();
	for(i = 0; i < M; i++){
		pid = fork();
		if(pid==0) {
			close(0);
			dup(fd);
			execlp("./replicant","./replicant",argv[2],(char*) 0);
			close(fd);
		}
	}
	int acum = 0;
	int slave = waitpid(-1, &status, 0);
	while ( slave > 0) {
		if(WIFEXITED(status)){
			int8_t code;
			code = WEXITSTATUS(status);
			exitcode = code;
			acum += exitcode;
		}
		else{
			printf("--> ERROR IN CHILD EXIT STATUS\n");
		}
		slave = waitpid(-1, &status, 0);
	}
	dummy_test(acum, ENIGMA1, TEAMNAME);
	exit(0);

}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include "dummy2.h"

void Usage(int n){
        printf("Usage: %d, on 0 < n < 2000\n",n);
}

void main (int argc, char *argv[]){
        int n,fd, bytesllegits;
        char buffer[2000];
        n=atoi(argv[1]);
        if(n>=0) if(n<=2000) Usage(n);
        bytesllegits = 0;
        dummy_init (buffer, n);
        bytesllegits = read(0,buffer,n);
        while(bytesllegits>0){
                dummy_comp(buffer,bytesllegits);
                bytesllegits = read(0,buffer,n);
        }
        dummy_exit();
}
```

COMANDES:

```
-@ubiwan:~/SIOP/lab2$ ./gendummy
Usage: ./gendummy [S|M|L], to create SMALL, MEDIUM or LARGE dummy file
-@ubiwan:~/SIOP/lab2$


gcc replicant.c -o replicant  -L. -ldummy2
gcc TT.c -o TT -L. -ldummy2
mariona@mariona/Escritorio/SIOP/Lab/Capitol-Lab1$ ./programaTT 2 3
```