Domanda **1**
Risposta non ancora data
Contrassegna domanda

In authentication systems, which property of a hash function is the least significant?

Scegli un'alternativa:
- Strong collision resistant.
- Computationally hard to identify two inputs x, y such that H(x) = H(y).
- Computationally fast.
- Computationally infeasible to find x such that H(x) = h.

Domanda **2**
Risposta non ancora data
Contrassegna domanda

How can you prevent SQL injection attacks?

Scegli un'alternativa:
- By dropping network connections to the database.
- By perturbing the raw data of the database.
- By perturbing the output returned to the user.
- By sanitizing users' input.

Domanda **3**
Risposta non ancora data
Contrassegna domanda

Choose the most appropriate answer. A discretionary access control can be represented with:

Scegli un'alternativa:
- Matrix structure.
- List structure.
- Both with Matrix Structure and List Structure.
- None of the above.

Domanda **4**
Risposta non ancora data
Contrassegna domanda

What are "canaries" used for?

Scegli un'alternativa:
- Detecting DDoS attacks.
- Detecting intruders.
- Encrypting packages.
- Detecting buffer-overflow attacks.

Domanda **5**
Risposta non ancora data
Contrassegna domanda

What is a common practice for attackers during DoS attacks?

Scegli un'alternativa:
- Injection codes.
- IP forgery.
- Buffer overflow commands.
- None of the above.

Second Part - Second Module

Is the program leak.c vulnerable to a buffer overflow of type stack smashing/control hijacking?
(If yes why, if no why)

Is the program leak.c vulnerable to a buffer overflow of type information exfiltration?
(If yes why, if no why)

Is the program vulnerable if stack protectors (canaries) are in place?
(If yes why, if no why)

Is the program vulnerable if ASLR is in place?
(If yes why, if no why)

Can not-executable stack prevent the return loop in loop3.c?
(If yes why, if no why)

Can canaries prevent the return loop in loop3.c?
(If yes why, if no why)

Let's make the Hypothesis that there is no "system" function call in libc: is it still possible to make a return to libc attack?
(If yes why, if no why)
Let's make the Hypothesis that there is no "system" function call in libc: Is it possible to make a return to libc attack that activates a shell?
(If yes why, if no why)
Let's make the Hypothesis that there is no "system" function call in libc: Is it possible to make a return to libc attack that activates a ROOT shell?
(If yes why, if no why)

Assume you know the address of a http server with the shellshock vulnerability and a myCommand CGI program available (in the default CGI-BIN folder).
Write the curl command that makes an http request that exploits the shellshock vulnerability and executes the "cat /etc/passwd" command redirecting output to a /tmp/log.txt file

leak.c	12 giugno 2021, 11:22
loop3.c	12 giugno 2021, 11:22

leak.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define SIZE 16
unsigned char *isThisVulnerable(unsigned char* src, int size)
{
	unsigned char* retval = (unsigned char*)malloc(size*sizeof(unsigned char));
	return memcpy((void*)retval, (const void*)src, (size_t)size);
}
int main(int argc, char **argv)
{
	int i = SIZE, k = 0;
	if(argc > 1)
	{
		i = atoi(argv[1]);
	}
	unsigned char buf[SIZE];
	for(k=0;k<SIZE;k++)
		buf[k] = 0;

	unsigned char *result = (unsigned char*)isThisVulnerable(buf, i);
	printf("%02X", result[0]);
	for(k=1;k<i;k++)
	{
		if(k%16 == 0)
			printf("\n");
		else if(k%4 == 0)
			printf("\t");
		printf("%02X", result[k]);
	}
	printf("\n\n");
}
```

loop3.c

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 0
int k = 0;
void loop()
{
	unsigned int buf[SIZE];
	unsigned int i = 0;
	printf("%d Now loop starts...\n", k++);

	if(k<1000)
	{
		loop();
	}
/*
*/

	printf("buf is at %X\n", (unsigned int)buf);
	for(i=SIZE + 0;i<SIZE + 10;i++)
	{
		printf("[%d] = %X\n", i, buf[i]);
	}

	*((int*)(&(buf[SIZE + 5]))) = &loop;
	*((int*)(&(buf[SIZE + 4]))) = &loop;
	//*((int*)(&(buf[SIZE + 3]))) = buf[SIZE + 3] - 4;
	//*((int*)(&(buf[12]))) = &loop;
	//*((int*)(&(buf[9]))) = &loop;
	//*((int*)(&(buf[8]))) = &loop;
	//printf("count:%X inbuf:%X i:%X buf:%X\n", &count, inbuf, &i, buf);
	//for(i=0;i<count;i++)
	//{
		//buf[i] = inbuf[i];
```

```
                //printf("%X %d\n", &(buf[i]), i);
        //}
        printf("Now loop returns...\n");
}

int main(int argc, char** argv)
{
        int val = 7;
        int k = 0;
        int dim = atoi(argv[1]);
        unsigned char theBuf[dim];
        if(argc > 2)
        {
                val = atoi(argv[2]);
        }
        for(k=0;k<dim;k++)
        {
                theBuf[k] = val;
        }
        printf("main = %X\n", (unsigned int)main);
        printf("loop = %X\n", (unsigned int)loop);
        loop(dim, theBuf);
}
```