# A blue and white sign with textSistemas de Segurança de Software

## Mestrado em Engenharia Informática

# Segurança de Software

## Mestrado em Informática

# Class Project: Experiments with Format Strings

Gustavo Henriques Nº 64361

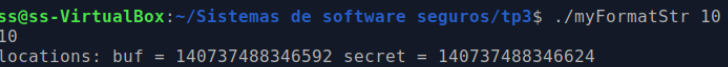
Leonardo Monteiro Nº 58250

Maria Figueirinhas Nº 46494

d) O programa define dois buffers, secret[20] e buf[32], contendo 20bytes e 32bytes de memória disponível respetivamente (linhas 1 e 2). O 1º strncpy copia para dentro de secret a string “secret!!”, tendo em conta o tamanho deste buffer (linha 4). O 2ºstrncpy copia para dentro de buf o argv[1], que é fornecido no terminal pelo utilizador (linha 5). Copia tendo em conta o tamanho de buf, até os 32bytes estarem preenchidos.

Nas linhas 7,8 e 9, existem vários printf. O primeiro imprime o conteúdo de buf, o segundo imprime uma nova linha e o terceiro imprime as localizações da memória onde buf e secret estão guardados.

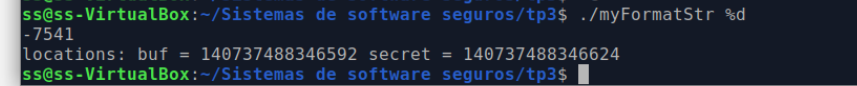
1. char secret[20];
2. char buf[32];
3. strncpy(secret, "secret!!", sizeof(secret));
4. strncpy(buf, argv[1], sizeof(buf));
5. printf(buf);
6. printf("\n");
7. printf("locations: buf = %lu secret = %lu\n", (unsigned long) buf, (unsigned long) secret);



e) Para imprimir o que está no rsi, basta dar como argv[1] “%d”, um format string que devolve o valor decimal.

1. strncpy(buf, argv[1], sizeof(buf));
2. printf(buf);

O valor “%d” será guardado dentro do buf. O prinft guarda no registo rdi o endereço de memória de buf e o seu conteúdo em rsi. Quando o programa lê rsi, está lá uma format string, ou seja a função prinft vai tentar interpretar esse argumento, e então imprime o conteúdo da localização onde buf está guardado. O resto do programa mantém-se inalterado.



f) Para imprimir os 8 bytes acima de rip, temos de fornecer como argumento uma série de format strings que preencham os registos de printf. Como há rdi e mais 5 registos (rsi, rdx, rcx, r8d, r9d), podemos escrever “%x%x%x%x%x” de modo a preencher esses registos. Quando estes registos são preenchidos, o programa guarda o que vier a seguir acima de rip, que é o que queremos imprimir.

Cada format string ‘simples’ imprime 4 bytes apenas, se queremos imprimir 8 bytes, teremos de adicionar ‘l’, long, que aumenta o que é impresso. Por este motivo, fornecemos como argumento após o que já foi escrito %lx.

O argumento completo mostra-se na imagem abaixo, com o respetivo output no terminal.



g) Segundo o código Assembly, o rip está 80 bytes abaixo de rbp e o registo de argc está a 68 bytes abaixo de rbp. Ou seja, temos de imprimir 12 bytes acima de rip. Podemos então escrever de modo a encher os registos de prinft (5 argumentos para 5 registos) e acrescentar mais 2 argumentos (8bytes + 4bytes). O final do último output será o argc (2, uma vez que é ‘nome do programa’ e ‘argumento’).



h) Segundo o código Assembly, o rip está 80 bytes abaixo de rbp e o registo de secret está a 48 bytes abaixo de rbp. Ou seja, temos de imprimir 32 bytes acima de rip. Podemos então escrever de modo a encher os registos de prinft (5 argumentos para 5 registos) e acrescentar mais 4 argumentos (32 bytes). O final do último output será o que está no quarto registo acima do rip (“secret!!!”, uma vez que é onde foi guardada a variável secret).

