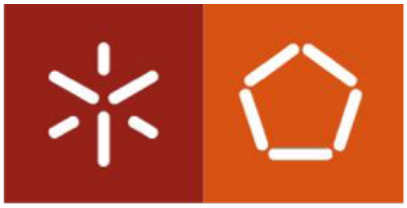

Universidade do Minho
Departamento de Informática
Mestrado Integrado em Engenharia Informática

Engenharia de Segurança

Aula 10

Diana Lopes, nº a74944
Gabriela Vaz, nº a74899

Conteúdo

Capítulo 1

Pergunta P1.1

Em todos os programas, são alocadas apenas 10 posições em memória para os valores de input. Verifica-se que, nos programas **.java** e **.py**, quando se excedem esses 10 valores, há uma mensagem de erro e não é permitido que sejam inseridos mais valores. No entanto, o programa **.cpp** continua a aceitar valores de input mesmo quando já excedeu os 10 valores que o *array* pode receber. Isto resulta na escrita de posições de memória aleatórias, podendo eventualmente corromper outros programas, caso essas posições de memória não estivessem livres.

Capítulo 2

Pergunta P1.2

Tanto no ficheiro **.java** como no ficheiro **.python**, quando tentamos inserir mais elementos do que o número dado como limite no código, o programa dá-nos um erro.
Já o ficheiro **.cpp** aceita como input um valor superior ao valor máximo apresentado no código. Por exemplo, o programa aceita que se insiram mais valores do que os que estão alocados para o *array* mas quando se pretende recuperar um valor armazenado que esteja numa posição do *array* superior ao limite apresentado no código, é retornado o endereço de memória onde o elemento está alocado. Este programa permite ainda que, por exemplo, se insiram 30 elementos (valor maior do que o limite alocado para o *array*) e se consulte o elemento que está na posição 40 (retornando, como dito anteriormente, a posição de memória).

Capítulo 3

Pergunta P1.3

1. **RootExploit.c**

   A vulnerabilidade encontrada é o facto de o programa não validar a *password* inserida. Assim, é possível uma inserir uma *password* aleatória e esta ser aceite, desde que o tamanho desta seja superior a 4 (tamanho do *buffer*).

   Para conseguir as confirmações das permissões, como já referimos anteriormente, basta colocar uma *password* aleatória, com um tamanho aleatório.

2. **0-simple.c**

   O objetivo é fazer aparecer a mensagem "You win!". Para isso, é necessário alterar o valor da variável **control** para um valor diferente de 0. Isso consegue-se inserindo-se uma *string* grande o suficiente que exceda o **buffer** e, assim, permita o acesso à variável **control**.

Capítulo 4

Pergunta P1.4

No programa ReadOverflow.c o número de bytes alocados para o *buffer* é 100. No entanto, é possível dar um valor superior a 100 e, assim, aceder a outras posições de memória que podem, eventualmente, ter conteúdo "sensível".

Capítulo 5

Pergunta P1.5

Ao correr o programa verificamos que a partir de um determinado tamanho de input começa-se a alterar o valor da variável *control*. Como a variável *buffer* está a ser definida depois de *control* faz com que seja alocada fora dos limites do próprio *buffer*. Assim, se as subtrairmos conseguimos separá-las.
Por fim, imprimimos o caracter quando o valor de *offset* (obtido através da subtração de ambas as variáveis) pedindo também para imprimir a variável *control*. É necessário escrever bytes na ordem contrária, uma vez que arquitetura representa os dados no formato *little-endian*.

Capítulo 6

Pergunta P1.6

Fazendo *debug*, encontra-se o endereço da função **win**. Depois converte-se o endereço da função para código ASCII e invertem-se os bytes (por causa do formato *little-endian*, como referido anteriormente). O código ASCII é somado com uma *string* de (pelo menos) 72 *bytes* (tamanho necessário para alterar a variável *fp*). Assim, é possível atingir o objetivo.