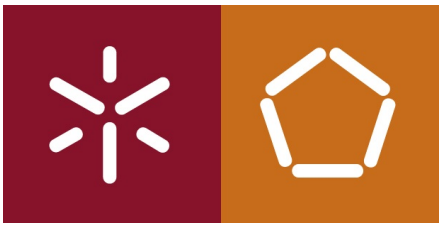


Engenharia de Segurança
Grupo 7
Aula 10

Bruno' - A74941
Diogo Gomes - A73825
Francisco Mendes - A75097

Abril 2018

1

- No programa de java ocorre uma exceção de ArrayIndexOutOfBounds devido ao controlo de memório existente no java que não deixa que hajam escritas fora dos limites de um array.
- Tal como o java o Python também implementa controlo de memória impedindo que sejam feitas escritas fora dos limites do array, respondendo com uma mensagem de IndexError.
- Em C++ não existe controlo de memória, deste modo ao tentar escrever na posição 10, o programa vai escrever no espaço de memória seguinte, podendo ocorrer várias ações, entre elas o segmentation fault.

2

- No programa de java ocorre uma exceção de ArrayOutOfBounds devido ao controlo de memório existente no java que não deixa que hajam escritas fora dos limites de um array.
- Tal como o java o Python também implementa controlo de memória impedindo que sejam feitas escritas fora dos limites do array, respondendo com uma mensagem de IndexError.
- Em C++ não existe controlo de memória, como neste caso a variável que é preenchida antes do array é a variável "i", o seu valor estará declarado acima do buffer. Se for pedido para guardar um número de caracteres maior que 10 que neste caso é o tamanho do buffer, o valor da variável "i"será alterada quando for escrito no indice 10 do array. Esta alteração pode levar a um loop infinito no caso de valores entre [11,19]. Valores superiores ou iguais a 20 produzem resultados imprevistos uma vez que reescrevem outra variáveis sem ser a variável "i". Se o valor inserido for superior ao tamanho da stack do programa, leva a que ocorra o erro de segmentation fault.

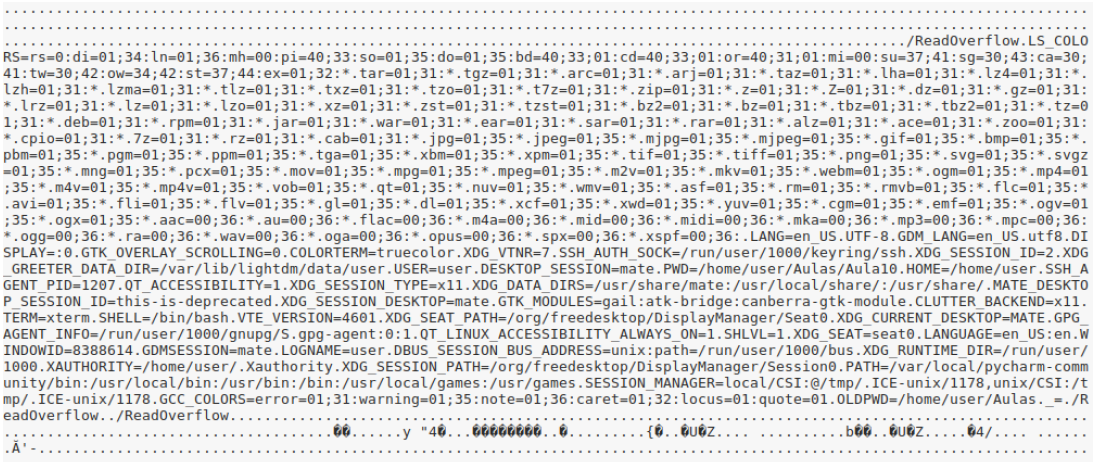
3

Ambos os programas apresentam a vulnerabilidade conhecida como **Buffer Overflow**.

- **RootExploit**: ao exceder o tamanho reservado do buffer para receber o input da password pode "corromper"a `flag` que valida a passwordud. Assim, apesar de se introduzir a palavra passe errada são dadas permissões de utilizador root.
- **0-simple:** à semelhança do programa anterior, ao exceder o tamanho do buffer que recebe o input pode-se alterar o valor da variável `control`, resultando na mensagem "YOU WIN!!!"

4

Neste programa, quando se declara que o tamanho da frase que se vai introduzir é relativamente grande, para além de imprimir lixo no final do buffer, foi possível aceder ao valor de algumas variáveis de ambiente:



Para produzir este output, declarou-se que se ia introduzir uma frase com o tamanho máximo que um inteiro pode tomar (2147483647) afim de obter mais informação.

5

Sabendo que 0x61626364 corresponde a "abcd"em ASCII, e como dados são guardados em memória no formato little-endian, é possível obter "dcba". Após obter o que se pretende, é necessário colocar esse valor na variável *control*, e pode-se fazer isso colocando um valor qualquer antes do texto, neste caso foi utilizado um conjunto de a's.

```
user@CSI:~/Aulas/Aula10$ ./1-match aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaadcba
You win this game if you can change variable control to the value 0x61626364'
Congratulations, you win!!! You correctly got the variable to the right value
user@CSI:~/Aulas/Aula10$
```

6

Depois de ter sido compilado o programa com a flag -g. Foi chamado o gdb com o executável gerado para descobrir o endereço da função win() através do comando "p win". O endereço retornado foi "0x555555554740", passando os valores em hexadecimal para caracteres através da tabela ACSII resulta a frase "UUUUG@". Na stack do programa ficam registados os seguintes endereços de cima para baixo: declaração fp, declaração buffer, return, base pointer, atribuição fp e atribuição do buffer. Para que seja chamada a função win será necessário escrever no return o endereço da função win. como os índices do buffer crescem de baixo para cima, o seu ultimo índice fica abaixo da atribuição da variável fp. Assim ao preencher o buffer se utilizarmos 72 caracteres seguidos de "@GUUUU"uma vez que a arquitetura é little endian, serão preenchidos 64 indices do buffer 4 bytes da declaração do fp, 4 bytes do base pointer, e finalmente será preenchido o endereço de return com o endereço da função win().

7

Este exercício assemelha-se ao exercício anterior no sentido em que é necessário exceder os limites de um buffer de tamanho 64 afim de aceder ao endereço de retorno da função `win`. O primeiro passo será o de descobrir o endereço da função `win` que é `0x5555555546f0`. Traduzindo este registo para código ASCII resulta na frase "UUUUFð"e concluia-se este exploit ao escreverem-se 72 carateres e acrescentando-se ao sufixo a frase na ordem inversa ("ðFUUUU")

uma vez que a arquitetura é little endian. Contudo, não se conseguiu alterar o endereço de retorno para o pretendido dado pois ð é um caracter ASCII especial e ocupa 2 bytes ao invés do convencional, tornando impossível alterar o endereço de retorno para `0x5555555546f0`.