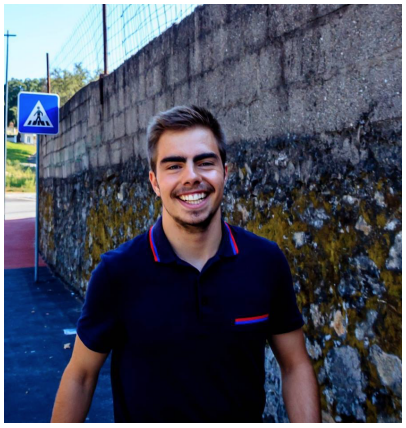
Escola de Engenharia
Universidade do Minho

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA
Mestrado em Engenharia Informática
Engenharia de Segurança

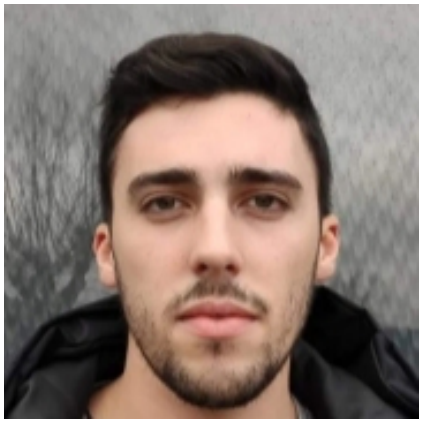
Aula 11

4 de Maio de 2020

Grupo 1



Ricardo Pereira a73577



Tiago Ramires pg41101

Braga, 8 de Maio de 2020

1. Integer Vulnerability

Question P1.1

O programa em questão recebe uma *string* como *input* do utilizador e não lhe faz qualquer tipo de tratamento. Esta string deveria ser um nome de um ficheiro, sendo apresentado como *output* fa função o seu tipo. Através da função *snprintf*, é copiado o conteúdo da *string input* para um , adicionando no início desse buffer, a palavra *file* que juntamente com a *string input* formará o comando que imprime na *bash* o tipo do ficheiro introduzido.

1

As vulnerabilidades que se verificam neste programa são, por exemplo, a falta de validação de *input* proveniente do processo-pai e a falta de validação de*input* quanto aos (meta)carateres aceites.

2

Execução de outros comandos

O não tratamento dos (meta)carateres constitui uma grande vulnerabilidade neste programa, uma vez que permite, para além da execução do comando *file*, a execução de tantos comandos quantos o utilizador quiser, bastando para isso utilizar, por exemplo, o separador "&&"para os diferenciar. No exemplo, a seguir, temos um *input* para o programa que vai devolver o conteúdo do ficheiro *filetype.c*.

```
./a.out "filetype.c && cat filetype.c"
```

Alteração da variável *PATH*

É possível manipular a variável *PATH* e alterar a informação que a mesma contém. Por exemplo, o comando *ls* tem uma função, mas alterando esta variável, era possível colocar o comando *ls* a ter outra função que não a mostragem das diretorias e ficheiros daquela pasta. A título de exemplo, criamos um programa simples chamado *malware*:

```
int main() {
        printf("Estou a ser executado em vez do comando file.\n");
}
```

Posto isto, demos a localização do programa à variável *PATH* e o programa passou assim a estar disponível para execução sem a necessidade de introdução da sua localização.

```
export PATH=$PATH:<caminho do nosso programa>
```

Finalmente, a execução do comando *malware* retornou o *output* esperado.

3

Caso o programa tivesse permissões *setuid root*, então o utilizador também as teria. Tal comportamento, poria em risco tudo aquilo que é possível fazer com as permissões de *root* do sistema operativo.

Question P1.2

O programa em questão foi feito em *Python* e para a validação dos carateres inseridos pelo utilizador utilizaram-se expressões regulares. Sendo assim adotaram-se as seguintes medidas:

- **valor a pagar** - apenas podem ser introduzidos algarismos e um ponto, no formato comum dos números decimais, sendo que a parte decimal pode ou não existir;
- **data de nascimento** - apenas é permitida a inserção de uma data no formato "DD-MM-AAAA", com algarismos e hífens e com a verificação do dia, mês e ano;
- **nome** - apenas são permitidos carateres do alfabeto, podendo ser introduzidos até oito nomes, tendo cada um um mínimo e um máximo de três e quinze carateres, respetivamente;
- **número de identificação fiscal (NIF)** - apenas são permitidos nove algarismos;
- **número de identificação de cidadão (NIC)** - apenas são permitidos oito, nove ou dez algarismos;
- **numero de cartão de crédito** - apenas são permitidos dezasseis algarismos;
- **validade** - apenas é permitida a inserção de uma data no formato "DD/MM", com algarismos e hífens e com a verificação do dia e do mês;
- **CVC/CVV** - apenas é permitida a inserção de 3 algarismos;

O programa chama-se *safe.py* e encontra-se na mesma diretoria que este relatório.