



UNIVERSIDADE DO MINHO

DEPARTAMENTO DE INFORMÁTICA

MESTRADO EM ENGENHARIA INFORMÁTICA

CRİPTOGRAFIA E SEGURANÇA DE INFORMAÇÃO

Engenharia de Segurança

Ficha Prática 12

Grupo Nº 3

Ariana Lousada (PG47034) Luís Carneiro (PG46541)
Rui Cardoso (PG42849)

7 de junho de 2022

1 Validação de Input

1.1 Pergunta 1.1

Analise o programa `readfile.c` que imprime no écran o conteúdo do ficheiro passado como argumento, a que acrescenta o sufixo `".txt"` de modo a garantir que só deixa ler ficheiros em texto.

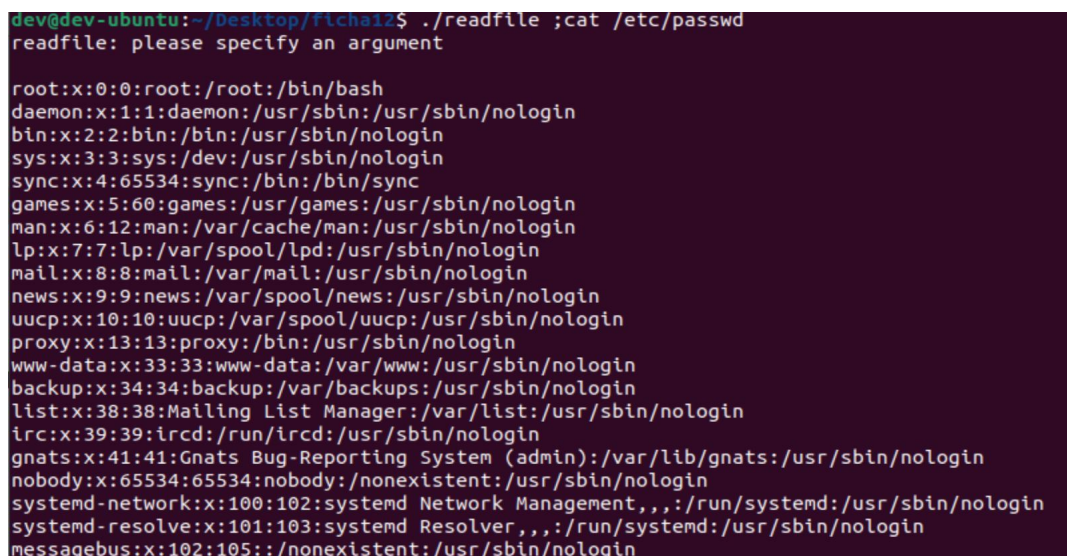
1.1.1 Existe pelo menos uma vulnerabilidade estudada na aula teórica de "Validação de Input" (em conjunto com outra que já estudou) que permite que o programa imprima ficheiros que não terminam em `".txt"`. Explique.

O método utilizado neste caso para ler o ficheiro `passwd` foi injeção de separadores. Ao inserir o metacaracter `';` como entrada de argumento é possível correr código arbitrário. Com isto podemos ler ou escrever informação em ficheiro que não era suposto.

Outro método para ler ficheiro que não terminem em `.txt` é a injeção do caracter `"\0"` que é interpretado como terminador de string. Ou seja, se passar como argumento `/etc/passwd\0.txt` já não vai ser lido. Isto pode ser aplicado para ler outro tipo de extensões.

1.1.2 Indique a linha de comando necessária para aceder ao ficheiro `/etc/passwd`.

A linha de comando usada foi `./readfile ;cat /etc/passwd`.



```
dev@dev-ubuntu:~/Desktop/ficha12$ ./readfile ;cat /etc/passwd
readfile: please specify an argument

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:102:105:/:/nonexistent:/usr/sbin/nologin
```

Figura 1: Linha de comando.

1.2 Pergunta 1.2

Desenvolva um programa (na linguagem em que tiver mais experiência) que pede:

- valor a pagar,
- data de nascimento,
- nome,
- número de identificação fiscal (NIF),
- número de identificação de cidadão (NIC),
- numero de cartão de crédito, validade e CVC/CVV.

Valide esse input de acordo com as regras de validação "responsável", apresentadas na aula teórica.

A linguagem escolhida foi C. O código pode ser consultado na totalidade no git. Para validar o input as seguintes condições tem de ser verificadas:

- Tipo
- Tamanho
- Intervalo
- Razoabilidade
- Divisão por zero
- Formato
- Dados obrigatórios
- Checksums

Começamos por verificar a variável *valor a pagar*:

```

164 int main(int argc, char **argv) {
165     char buf[BUF_LEN];
166     float f;
167
168     // valor a pagar
169     // documentação de strtod em
170     // https://www.ibm.com/docs/en/zos/2.4.0?topic=functions-strtod-convert-character-string-float
171     printf("Valor a pagar: ");
172     //fgets(buf, BUF_LEN, stdin);
173     strcpy(buf, "22.23");
174     f = strtod(buf, NULL);
175     if(f == -HUGE_VALF || f == HUGE_VALF || f == 0){
176         printf("Erro a inserir o valor a pagar\n");
177     } else {
178         printf("%.2f\n", f);
179     }
180 }

```

Figura 2: Valor a pagar.

Através do `strtod()` convertemos o valor para float e garantimos que este se encontra correto. De seguida a data de nascimento é verificada.

```

181 // data nascimento
182 printf("Data de nascimento (DD/MM/AAAA): ");
183 //fgets(buf, BUF_LEN, stdin);
184 strcpy(buf, "22/3/1990");
185 if(!validate_birth_date(buf))
186     printf("Data de nascimento errada!\n");
187 else printf("%s\n", buf);

```

Figura 3: Data de nascimento

A função `validate_birth_date()` é chamada e é passado o `buf`.

```

bool validate_birth_date(char* buf){
    char* str;
    strcpy(str, buf);

    // split em /
    char *delim = "/";

    int n=0; // guarda o número atual (dia/mês/ano)
    int data[2]; // guarda os numeros da data para facilitar a sua validacao
    int i=0; // i vai guardar o número de números divididos por /
    bool error=false;

    char *ptr = strtok(str, delim); // DIA
    while(ptr != NULL)
    {
        //printf("%s\n", ptr);
        if(string_to_int(ptr, &n))
            data[i]=n;
        else {
            printf("Erro a converter a string %s!\n", ptr);
            return false;
        }

        ptr = strtok(NULL, delim);

        i++;
        if(i>=3) // se contiver muitos números
            break;
    }

    // valida o número de números dentro da data
    if(i!=3) {
        printf("A data apenas pode conter 3 números\n");
        return false;
    }

    // validar a data em si
    // por exemplo se a data errada 30/02/.. foi introduzida
    if(!validate_date_year(data)){
        printf("Ano incorreto\n");
        return false;
    } else if(!validate_date_month(data)){
        printf("Mes incorreto\n");
        return false;
    } else if(!validate_date_day(data)){
        printf("Dia incorreto\n");
        return false;
    }

    return true;
}

```

Figura 4: Validate Birth Date

Esta função com a ajuda de outras 3 funções (validate_date_year(), validate_date_month(), validate_date_day e is_leap_year) valida o input para a data de nascimento.

```

37 // DD/MM/AAAA
38 bool validate_date_year(int data[]){
39     // buscar o ano atual para perceber
40     // se o utilizador não tem mais de
41     // 100 anos
42     time_t t = time(NULL);
43     struct tm tm = *localtime(&t);
44     int current_year = tm.tm_year + 1900;
45
46     if((current_year - data[2])<100)
47         return true;
48     return false;
49 }
50
51 // DD/MM/AAAA
52 bool validate_date_month(int data[]){
53     if(data[1]>=1 && data[1]<=12)
54         return true;
55     return false;
56 }
57
58 // DD/MM/AAAA
59 bool validate_date_day(int data[]){
60     int day=data[0];
61     int month=data[1];
62     int year=data[2];
63
64     // válido para todos os meses
65     if(day<1 || day>31)
66         return false;
67
68     int max_day;
69     if(month==2) // fevereiro
70         if(is_leap_year(year)) max_day=29;
71         else max_day=28;
72     else // meses com 31 dias
73         if(month==1 || month==3 || month==5 || month==7 ||
74            month==8 || month==10 || month==12)
75             max_day=31;
76         else // meses com 30 dias
77             max_day=30;
78
79     if(day>max_day) return false;
80     return true;
81 }

```

Figura 5: Funções usadas por ValidateBirthDate

```

13 // retirado de https://www.programiz.com/c-programming/examples/leap-year
14 bool is_leap_year(int year){
15     // leap year if perfectly divisible by 400
16     if (year % 400 == 0) return true;
17     if (year % 100 == 0) return false;
18     if (year % 4 == 0) return true;
19     return false;
20 }

```

Figura 6: is_leap_year

O próximo input a validar é o NIF.

```

189 // NIF
190 int n=0; // para guardar o nif depois de estar convertido
191 printf("NIF: ");
192 //fgets(buf, BUF_LEN, stdin);
193 strcpy(buf, "135785413");
194 if(!validate_nif(buf, &n))
195     printf("NIF invalido!\n");
196 else
197     printf("%d\n", n);
198

```

Figura 7: Validação NIF

Esta função faz uso das funções check_nif_digits() e validate_nif.

```

136 // adaptado de https://pt.wikipedia.org/wiki/N%C3%BAmero_de_identifica%C3%A7%C3%A3o_fiscal#Obter_d%C3%ADgito_de_controlo
137 int check_nif_digit(char nif[]){
138     //soma = sum([int(digito) * (9 - pos) for pos, digito in enumerate(string_num)])
139     int soma=0;
140     for(int i=0; i<NIF_LEN-1; i++){
141         //printf("%d * %d\n", nif[i] - '0', NIF_LEN - i);
142         soma += (nif[i] - '0') * (NIF_LEN - i);
143     }
144     int resto = soma % 11;
145     if(resto==0 || resto==1)
146         return 0;
147     return 11-resto;
148 }
149
150 bool validate_nif(char buf[], int *nif){
151     if(strlen(buf) != NIF_LEN) return false; // ver se tem tamanho 9
152     if(!string_to_int(buf, nif)) return false; // ver se apenas tem numeros
153
154     //printf("NIF is %d\n", nif);
155     int check_d = check_nif_digit(buf);
156     printf("Check digit e %d.\n", check_d);
157
158     //compare checkDigit with the last number of NIF
159     if(check_d == (buf[NIF_LEN-1] - '0'))
160         return true;
161     return false;
162 }

```

Figura 8: Validate NIF