



Ciências  
ULisboa

Sistemas de Segurança de Software

Mestrado em Engenharia Informática

Segurança de Software

Mestrado em Informática

## Class Project: Experiments with Fuzzer

Gustavo Henriques Nº 64361

Leonardo Monteiro Nº 58250

Maria Figueirinhas Nº 46494

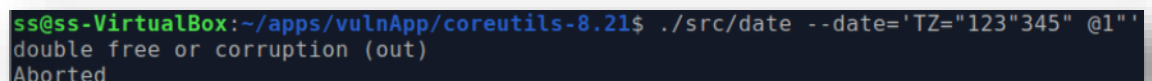
## 2. Fuzzing GNU Coreutils

### a) Checking that the vulnerability exists

Neste exercício começamos por correr os comandos que o enunciado mostrava. De seguida, experimentamos alguns dos comandos dados e depois, para explorar a vulnerabilidade metemos o seguinte comando:

```
./date --date='TZ="123"345" @1''
```

Com isto o terminal deixou de se comportar como devia, como é possível ver na imagem abaixo, concluindo assim que causamos um crash:



```
ss@ss-VirtualBox:~/apps/vulnApp/coreutils-8.21$ ./src/date --date='TZ="123"345" @1''
double free or corruption (out)
Aborted
```

### b) Fuzz the date program

Nesta alínea fizemos os comandos que estavam no enunciado e o fuzzer começou a correr, como vemos na imagem:

```
american fuzzy lop 2.57b (date)

process timing |-----| overall results
run time      : 0 days, 0 hrs, 2 min, 3 sec    cycles done : 0
last new path : 0 days, 0 hrs, 0 min, 4 sec    total paths : 438
last uniq crash : 0 days, 0 hrs, 0 min, 35 sec  uniq crashes : 3
last uniq hang  : none seen yet                uniq hangs  : 0
-----|-----|-----|
cycle progress |-----| map coverage
now processing : 206 (47.03%)                  map density : 0.32% / 1.45%
paths timed out : 0 (0.00%)                    count coverage : 4.19 bits/tuple
-----|-----|-----|
stage progress |-----| findings in depth
now trying     : interest 8/8                  favored paths : 79 (18.04%)
stage execs    : 165/166 (99.40%)              new edges on : 124 (28.31%)
total execs    : 341k                          total crashes : 223 (3 unique)
exec speed     : 2975/sec                      total tmouts  : 2 (2 unique)
-----|-----|-----|
fuzzing strategy yields |-----| path geometry
bit flips      : 14/2688, 5/2654, 4/2586        levels      : 3
byte flips     : 0/336, 2/302, 3/242            pending     : 405
arithmetics    : 20/18.7k, 0/1408, 0/54         pend fav    : 50
known ints     : 4/1645, 7/7406, 2/9361         own finds   : 437
dictionary     : 0/0, 0/0, 0/0                 imported    : n/a
havoc          : 379/290k, 0/0                  stability   : 100.00%
trim           : 18.64%/81, 0.00%
-----|-----|-----|
[cpu:297%]
```

### c) Confirm crash

Neste exercício fomos buscar um dos crashes e testar se realmente dava um crash. Como podemos ver na imagem abaixo, o crash aconteceu mesmo:

```
ss@ss-VirtualBox:~/apps/vulnApp/coreutils-8.21/src$ ./date --date='TZ=""'
double free or corruption (out)
Aborted
```

### d) Locate the vulnerability

Para este exercício corremos o comando `gdb ./date` no terminal para abrir o `gdb`. Dentro do `gdb` fizemos `run --date='TZ="ETZ="1"'`. Depois disto obtemos o crash dizendo “double free or corruption (out)” e, com o comando `where` percebemos que o crash ocorreu quando o ficheiro `date.c` tenta chamar a função `parse_datetime` que por sua vez no ficheiro `parse_datetime.y` irá tentar fazer `free`. O crash está a acontecer porque estamos a tentar libertar memória que já foi libertada.

### e) Determine if the "unique" crashes are really "unique"

Aqui tentamos o comando `run -date='TZ="m\\E"1="1"1'`. Com este comando obtivemos exatamente o mesmo erro. O que significa que apesar de se chamarem unique crash todos estão a explorar a mesma vulnerabilidade.

#### **f) Use AFL\_HARDEN to check for bugs**

Para este exercício usamos o AFL\_HARDEN para tentar encontrar diferentes tipos de crashes que explorassem diferentes vulnerabilidades. Para isso fomos correr o mesmo comando no gdb que fizemos anteriormente e reparamos que a vulnerabilidade explorada foi a mesma.