

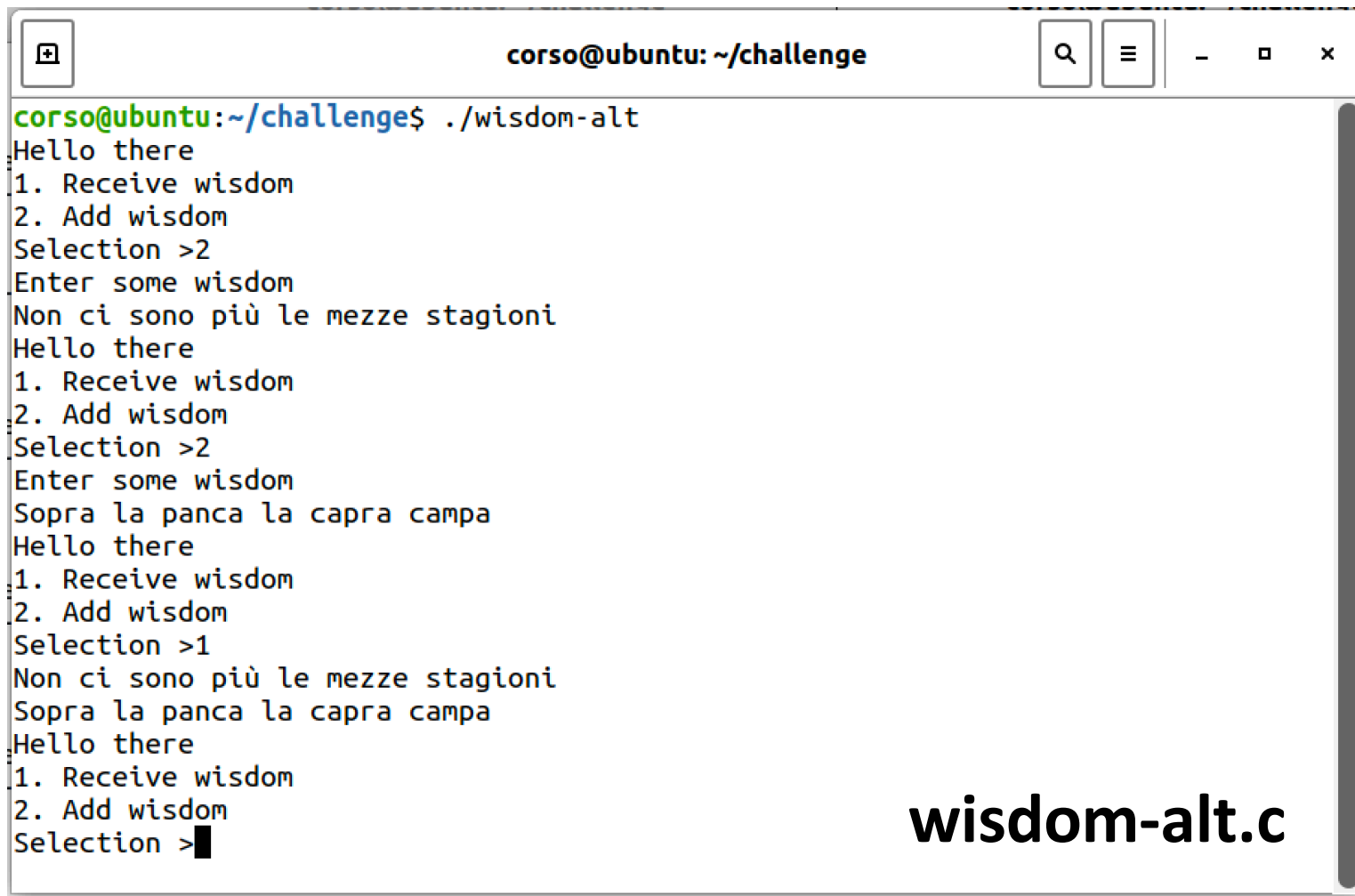
Software Security

Domenico Cotroneo

Roberto Natella



Challenge – buffer overflows



A terminal window titled 'corso@ubuntu: ~/challenge' showing the execution of the 'wisdom-alt' program. The program prompts the user to 'Hello there' and presents a menu with two options: '1. Receive wisdom' and '2. Add wisdom'. The user selects '>2' and enters the text 'Non ci sono più le mezze stagioni'. The program then prompts 'Hello there' again, and the user selects '>2' and enters 'Sopra la panca la capra campà'. This sequence repeats once more, with the user selecting '>1' and entering 'Sopra la panca la capra campà' in the final iteration shown. The program is named 'wisdom-alt.c'.

```
corso@ubuntu:~/challenge$ ./wisdom-alt
Hello there
1. Receive wisdom
2. Add wisdom
Selection >2
Enter some wisdom
Non ci sono più le mezze stagioni
Hello there
1. Receive wisdom
2. Add wisdom
Selection >2
Enter some wisdom
Sopra la panca la capra campà
Hello there
1. Receive wisdom
2. Add wisdom
Selection >2
Enter some wisdom
Sopra la panca la capra campà
Hello there
1. Receive wisdom
2. Add wisdom
Selection >1
Non ci sono più le mezze stagioni
Sopra la panca la capra campà
Hello there
1. Receive wisdom
2. Add wisdom
Selection >█
```

wisdom-alt.c

Challenge – buffer overflows

- Quali sono le due vulnerabilità di buffer overflow nel programma?
- Come fare a forzare la chiamata di queste funzioni?
 - pat_on_back
 - write_secret

Challenge - suggerimenti

- Una delle vulnerabilità è legata all'array globale *"ptrs"*...
 - Si provi a inserire un valore diverso da 1 o 2!
 - Quali sono gli indirizzi delle variabili *buf*, *ptrs*, *p*, e delle funzioni?
 - Prima di avviare il programma con *"run"*, impostare un breakpoint prima o dopo la *read()* (*"break wisdom-alt.c:97"*)
 - Stampare con *"print nomevar"*, proseguire con *"next"* (singola istruzione) oppure *"continue"*
- Per sfruttare *"ptrs"*, ricordarsi che la sintassi in C *"array[i]"* equivale a *"array + i*sizeof(array[0])"*

Challenge - suggerimenti

- La seconda vulnerabilità è un classico stack overflow su `gets()`
 - Per scrivere sul buffer, serve prima inviare la stringa "2\n"
 - La prima `read()` del programma legge in totale 1024 caratteri
- Per cui, il formato del payload è più complesso, es.:

```
$ python -c 'import sys; sys.stdout.write("2\n"+"A"*1022)'> payload_search
```

```
$ cat pattern_payload >> payload_search
```
- Per analizzare il contenuto dello stack durante l'attacco
 - usare ancora breakpoint ("*break wisdom-alt.c:63*")
 - avanzare con "*next*" e "*stepi*" (avanza di una singola istruzione assembly, per analizzare l'uso dello stack durante l'istruzione C di "*return*")