Format String Lab

Computer Security

Task 1

- Crashing the program
  - Hints
    - %s tries to read values by dereferencing memory
    - Dereferencing NULL or 0 is illegal
- Printing value from heap
  - Hints
    - Try putting secret[1]'s address onto the stack
    - BIG HINT: Try getting secret[1]'s address into int_input

Task 1

- Modifying secret[1]
  - Similar to reading secret[1]. Try using %n.
- Writing pre-determined value into secret[1]
  - Use more characters or less characters in printf

Task 2

- Hint:
  - Use input redirection to provide input to the program

gdb

- Gnu debugger
- Debugger for many languages (C/C++ included)
- Allows you to inspect what a program is doing at a certain point in its' execution

Steps to debug

- Compile using:
  `gcc prog.c –o prog` **`-g`**
- Run program in gdb using:
  `gdb OR gdb prog`
- You will get a prompt that looks like this:
  `(gdb)`
- If you didn't specify a program to run, load it in by using:
  `(gdb) file prog`

gdb

- gdb has an interactive shell and has a command history (up and down arrow keys), can auto complete words (with the TAB key) and some other features
- If you need help with a particular command, use “help”
  `(gdb) help <command>`

Basic gdb commands

- To run the program, use:
  `(gdb) run OR (gdb) r`
- To set breakpoints, use:
  `(gdb) break prog.c:10 OR`
  `(gdb) b prog.c:10`

  WHERE
  prog.c – filename to break at
  10 – Line number
- To break at a particular function, use:
  `(gdb) break printf`

Basic gdb commands

- Continue, step and next:

```
(gdb) continue
Moves to next breakpoint
(gdb) step
Execute next line of code
(gdb) next
Steps over functions as if they
were 1 line of code
```

Basic gdb commands

- Print and x:

```
(gdb) print var_name
Prints the value of the variable

(gdb) print/x var_name
Prints the value in hexadecimal

(gdb) x 0xaddress
Prints the value at the address

(gdb) x/n 0xaddress
Prints values from address to
address+(n words)
```