CSE 351 Lab 3 – Smoke

`$ gdb bufbomb`

`(gbd) list 136`

`(gdb) break 136`
- Or somewhere around the call to `Gets()`

`(gdb) run -u <UWnetID>`
- Substitute your UW Net ID

`(gdb) next`
- Until you enter input
- Enter a bunch of the same character (e.g. 'f's = 0x66 in ASCII or '3' = 0x33 in ASCII)
- Recommended that you use +/- 1 from a multiple of 8 to demonstrate how GDB displays bytes

`(gdb) x /5gx buf`
- Examines the entire buffer (5 "giant words" – 8 bytes each in GDB – in hex format); find your input

`(gdb) print &buf`

`(gdb) print $rsp`
- Notice that `buf` is at the top of the stack

`(gdb) info frame`
- Find the saved return address ("saved rip") and where it is located

`(gbd) x /10gx $rsp`
- Prints out the stack; find the saved `%rip`
- Calculate how many bytes of padding are necessary:
  7 blocks * 16 hex digits per block = 112 hex digits of padding

`(gbd) print smoke`
- This will give you your target address – the one you want to overwrite the return address with

Exit GDB and open `smoke.txt` in a text editor to add padding and target address (little endian!!!)
- Repeating characters
  - vim: `<len>i<sequence>C-[`
    - e.g. `5, 6, i, 3, 2, <space>, Ctrl-[` will insert hex digits for 56 ASCII '2' characters
  - emacs: `C-x ([seq]C-u[len]C-x)`
- Will work with or without spaces; with space might be easier for students to understand

`$ ./sendstring < smoke.txt > smoke.bytes`

Open `smoke.bytes` in a text editor to show what it looks like (this will not be entirely readable)
- Can open hex mode in `vim` (`%!xxd`) or emacs (`M-x hexl-mode`)

`$ gdb bufbomb`

`(gdb) break 136`
- Or whichever line you broke on before

`(gdb) run -u <UWNetID> < smoke.bytes`

`(gdb) next`

`(gbd) x /10gx $rsp`
- Notice the return address has changed from before

Let it continue running... smokin'!

> **Important:** *Every* time you change `<file>.txt`, you will need to use `sendstring` to recreate `<file>.bytes`.
>
> You *pass* in `<file>.bytes` to `./bufbomb` but you *submit* `<file>.txt`