Tutorial
Hexedit Hacking Demo and Static Analysis

Anton Gerdelan <gerdela@scss.tcd.ie>

Hacking and Data Security

- <u>Educated guess</u> expected structure / data format (code is data too)
- Plain text (case-sensitive) search or specific value search in hex
  - is there a default starting value for something?
  - programmers don't want to mess-up - ascii strings will be used instead of secret codes
  - program may be compiled with debugging symbols still in it…

Exploits

- Usually can't make a string <u>longer</u>
  - Can make a string shorter though - '\0'
- Hijack the program or reverse-engineer a new dummy program?
- Keep this stuff in mind for your own programs…
  - are you being careful enough with private or financial customer data?
- Lots of tutorials around for more sophisticated methods

Static Analysis

- analyse your code without compiling (as such)
- look for things compiler misses
- access array out of bounds
- invalid memory access (some types)
- warn about other bug-prone code
  - "*it worked on my machine*"
- can catch the **dread "Heisenbug"**

Tools

- **Lint** or a "linter" - fluff remover
  - (1979, Bell labs - based on a C compiler)
- **Cppcheck** - free software - Daniel Marjamäki
  - C and C++, including some template support
  - https://github.com/danmar/cppcheck
  - also in most repositories (apt-get, brew etc)
- **scan-build** (part of Clang)
  - I use this most often
  - also built into Xcode IDE I believe
- some other IDEs have one built in (google around for your favourite IDE)

```
scan-build gcc -o demo main.c -g

scan-build make
```

Static Analysis

- Usually just a compiler that is re-written to provide more information
    - At the expense of longer error-check time
- Run as part of your build script - or just occasionally
- Look up the types of errors they can catch in the manual/website
    - **Try them all!**