Introduction

There are more conferences for Hackers than conferences for safety. That is the problem.

Basic goals:
- Re-explain the basic ecosystem of software from a security perspective
- give you enough vocabulary to pass internship questions
- Dispell misconceptions about development security

Advanced goals:
- modern mitigation and development techniques
- introduction to source-code review and auditing (from a security perspective)

Prerequisites

Settings limits:
- done mostly C
- few problems which are not C-specific

The fine print:
- all your fancy languages have C/C++ Runtime
- Unix has a fine security model

Follow @ internetofshit on twitter

How to pass the exam

- access to lecture notes
- you should be able to demonstrate that you understand the term by explaining it in your own words
- give concrete examples
- create your own examples
- imagine internship interview
- there will be source code sample to audit (and fix)
- it won't be 100% clean
- it won't be exactly like "standard Epita code"
- security issues to fix are nasty ones
- write in proper English
- beware of attendance, there might be a pop quizz

The development cycle

- classical shops write specs
- ... and have devs who implement them
- ... and db experts who write databases
- ... and system engineers who work on deployment
- ... and testers
- ... and security auditors
- This does not work

Specialization is for insects Robert A. Heinlein

Problem: everyone should be involved in security & db expert should be involved in database security

- if the auditors find a bug: sometimes it's because the design is wrong
- auditors can't catch it all so devs must know about good practices

Tester vs Devs

- you don't want to pit testers vs devs
- a good tester is invaluable
- ... document and fix bugs

Database experts:
- a lot of "database experts" don't even know about SQL injection (don't get me started on PhP)

Release!

- so you get to "ship a Release" end software product: V5.0
  ... that's not always the end
- are you the vendor?
- ... not the case for Unix distro

Branch and Support

- Before release: branched for that version say: 5.0 beta
- Resources devoted to 5.0
- After release: keep on current
- Residual Resources devoted to 5.0

A bug:
- you got to fix it... and possibly ship 5.1
  but wait that means testing
  - what about branch 4? and 3?
  - End of life for a product (EOL)
  - Extended support release (ESR)

Security bugs

- a bug is not a security hole
- Most attacks are based on a series of bugs
- We want defense in depth
- Fixing one bug stops the attack!
- An attack is also called a exploit
- Software has vulnerabilities

Who did it:
- Developer found the bug
- External user found the bug
  ... Recognized as a security issue?
  ... External user nice or not?
  - proving it's a security issue?
  - being pro-active about it
  - Fixing it without letting the bad guys know

Vendors and Open-Source and...

- was Reported on bugtraq
- ... multiple times
- CVE: Common Vulnerabilities and exposures

Bugtraq. website to report bug

Timely releases

- Don't Release on Friday
- Accept: for vendors
- Have a "secure" channel for bugs
- Worst case scenario: ZERO days

Misconception

- It's too complicated it won't be exploitable
- The IE5 URL overflow
- Because it's encoded as 16 bit character

Not a security issue

- Software components get reused all the time
- Plan to be successful

Open Source vs Closed Source

- Closed Source is not more secure
- Lots of people know how to reverse-engineer
- The "sweep under the carpet" effect

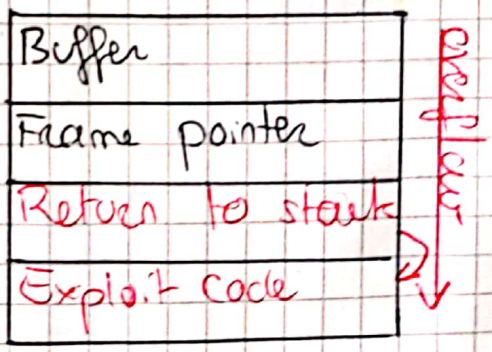
Example: Crafting exploits from Windows Update

I don't do bugs · it takes one bug
· Everything is exploitable

Buffer overflow

What's this?

```
void f(){
  char buffer[70];
  ...
  gets(buffer); //problematic line
}
```



gets(...) has been known for 20+ years to be unsafe
· has been fixed only in C++11
· use fgets(...) or getline(...) instead