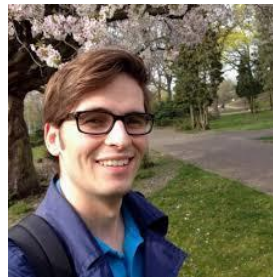


Software and Web Security

deel 1

About this course: people

- Pol van Aubel
- Ko Stoffelen
- Aaron van Geffen
- Erik Poll
- Peter Schwabe



About this course: topics & goals

- Standard ways in which software can be exploited
 - understanding how such attacks work
 - understanding what makes these attacks possible
 - doing some attacks in practice
- Root cause analysis: why are things so easy to hack?
- This involves understanding
 - programming languages, compilers, and operating systems, and the abstractions that they provide
 - the languages, representations, and interpretations involved
 - the potential for trouble – in the form of software vulnerabilities - that all this introduces

Software and Web Security - part 1 & 2

- part 1
 - security problems in machine code, compiled from C(++) sources (as usual), running on standard CPU and operating system

- part 2
 - security problems in software for the web, using web browsers and web applications, and typically some back-end database.

Prerequisites

- Imperatief Programmeren
 - we won't use C++, but C
 - biggest change: using `printf` instead of `>>` ?
- Processoren
 - what is the functionality that a typical CPU offers, on which we have to run our software written in higher-level languages?

Lectures & lab sessions

- 7 lectures and 7 lab sessions
- Lab sessions Mondays 8:45-10:30 in terminal room HG00.075
- Lectures Tuesdays 13:45-15:30 in Linnaeus 4
- All course material will be on
<http://www.cs.ru.nl/~erikpoll/sws1>

Lab exercises

Weekly lab session with weekly programming/hacking exercise

- *Exercises to be done in pairs*
- *Doing the exercises is **obligatory** to take part in the exam;*
- *Exercises will be lightly graded to provide feedback,
with **nsi-regeling**:
*you can have only one exercise niet-serieus-ingeleverd**
- But beware: exercises of one week will build on knowledge & skills from the previous week
- Also: turning up for the lab sessions might be *crucial* to sort out practical problems (with C, gcc, Linux, ...)

Lab exercises

We use

- C as programming language, not C++
- Linux from the **command line** aka **shell**
- the compiler **gcc**

So **no fancy graphical user interfaces (GUIs)**
for the operating system (OS) or the compiler

Why?

- GUIs are nice, but **hide** what OS and compiler are doing
- the command line is clumsy at first,
 - using commands instead of pointing & clickingbut gives great power
 - we can write **shell scripts**: programs that interact with the OS