Secure Programming (06-20010)
Chapter 1: Introduction

Christophe Petit

University of Birmingham

Before we start : Recording

- Every lecture is recorded and goes on Canvas
- Visible to University students only
- No video recording of the audience
- Your voice when asking questions might be on the recording

Secure Programming (06-20010)
Chapter 1: Introduction

Christophe Petit

University of Birmingham

The Importance of Secure Programming

Why do programmers write insecure code ?

- No customer demand for security
  - Secure code requires more time and money
  - Security measures may affect usability
  - Many users don't care about security
- Lack of security awareness and knowledge
  - Understand the threat model
  - Know at least the main security bugs
  - Use existing tools to help you
  - Keep updated at relevant places

Why do programmers write insecure code? (2)

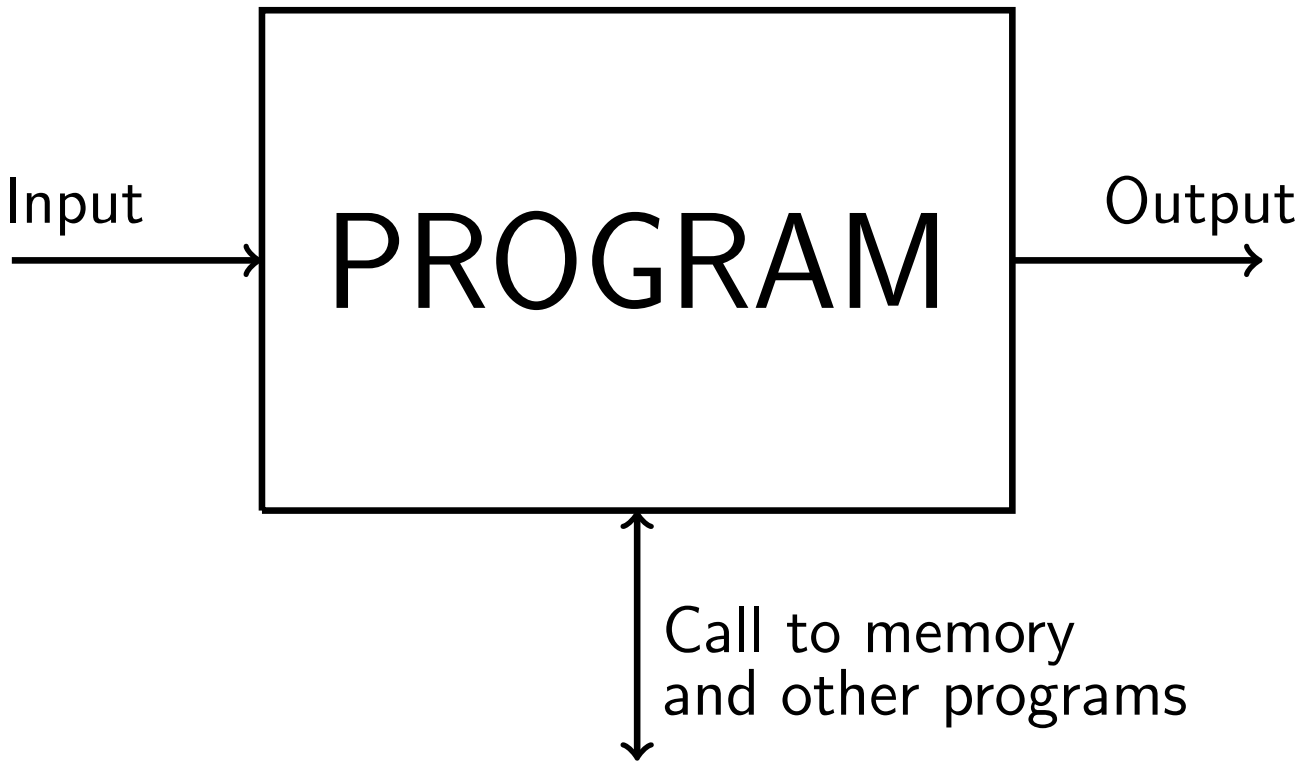
- Writing secure code is hard
  - Especially C/C++ programs
  - Especially for multi-user systems
  - Cannot control all the code you use or will use
- Lazyness
  - "Let's ignore all the warnings"
  - "If it works today, it should always work"

A quote from John F. Kennedy

There are risks and costs to a program of action - but they are far less than the long range cost of comfortable inaction.
- John F. Kennedy [1]

1. May 1961, talking about employment, equality, democracy, peace

“Secure programming” : Program ?

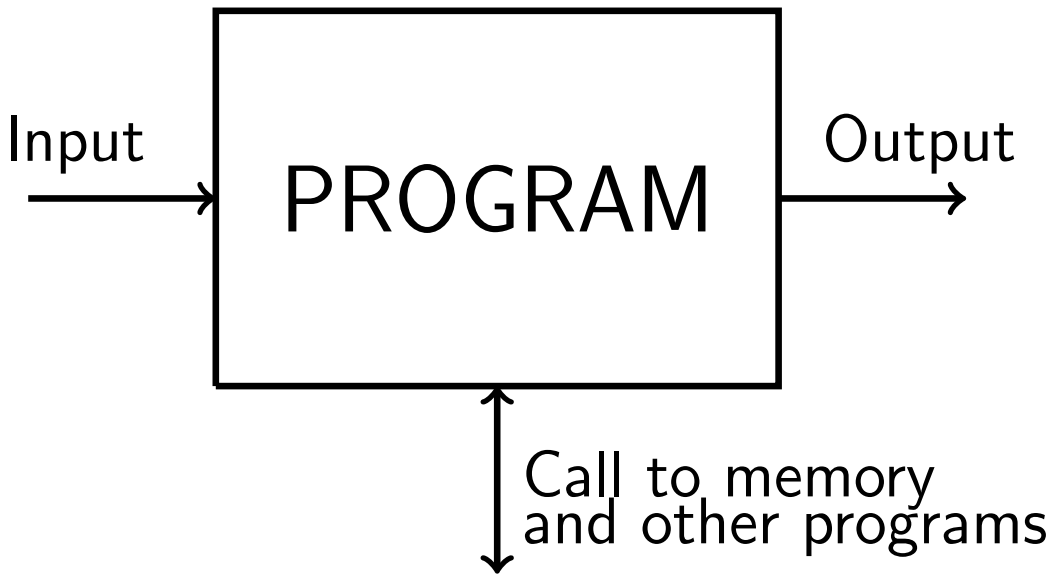


- In this course : some C, Java, Python, PHP, Shell, SQL

"Secure programming" : Secure ?

- Confidentiality : protect your data from unauthorized reading
- Integrity : protect your data from tampering
- Availability : data must be available to legitimate users (prevent denial-of-service)
- Authentication : check the identity of users and processes

"Secure programming"



- Know specific threats related to inputs, code, output, memory, etc
- Apply techniques and tools to protect against them

Need for secure programming

- Viewers of data coming from untrusted sources
- Programs with administrative privileges
- Web applications
- *setuid/setgid* programs in Unix
- ...

Secure Programming in the Master

MSc Cyber Security [504B]

Compulsory Modules to a total of 120 credits

Code	Title	Autumn	Spring	Summer
06-20008	Cryptography	10	-	-
06-28214	Designing Secure Systems	10	-	-
06-30016	Forensics and Malware Analysis	-	10	-
06-29637	Network Security (Extended)	-	10	-
06-18159	Project (CompSci - MSc)	-	-	60
06-20010	Secure Programming	10	-	-
06-28213	Secure System Management	10	-	-

Optional Modules to a total of 60 credits

Code	Title	Autumn	Spring	Summer
06-30017	Advanced Cryptography	-	10	-
06-15255	Compilers & Languages (Extended)	10	-	-
06-28206	Computer-Aided Verification (Extended)	-	10	-
06-28217	Hardware and Embedded Systems Security	-	10	-
06-19009	Individual Study 2	10	10	-
06-20233	Intelligent Data Analysis (Extended)	-	10	-
06-25689	Mobile & Ubiquitous Computing (Extended)	-	20	-
06-26950	Networks (Extended)	20	-	-
06-26952	Operating Systems (Extended)	20	-	-
06-28216	Penetration Testing	-	10	-
06-27822	Security Research Seminar	-	10	-

Why are you here today ?

- Tell me about your motivations
- Homework (unmarked) : email C.Petit.1 AT bham.ac.uk
  "Secure Programming will probably (not) be useful to me because . . . "

Our Learning Objectives

- Security awareness : understand the threats
- Principles of Secure Programming
- Most common classes of security bugs, why they occur, how to avoid them
- Hands-on practice with secure programming
- Pointers to learn beyond and keep up-to-date

Learning Methods

- Regular lectures : general principles, major vulnerabilities, why they occur and how to prevent them
- Computer labs : hands-on practice on selected topics
- Formative assignment (Weeks 4-6)
- Group exploration of a topic of your choice
- Self-learning : use references and web ressources !

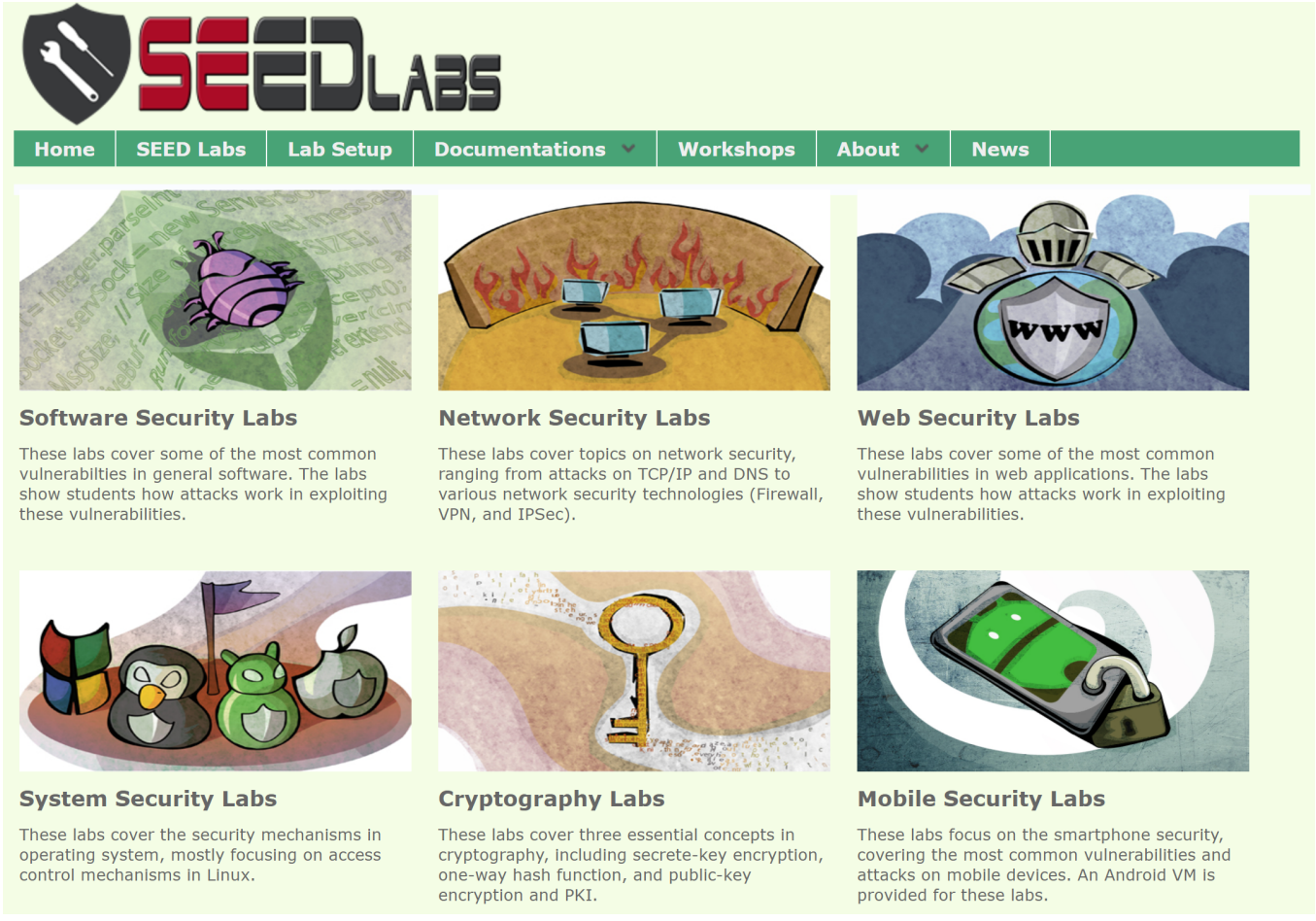
Lectures Content (tentative)

1. Introduction
2. General principles
3. Code injection (SQL, XSS, Command)
4. HTTP sessions
5. Unix Access Control Mechanisms
6. Race conditions
7. Integer and buffer overflows
8. Code review

Computer labs

- Labs are as important as lectures for this course !
- Our labs will be based on SEED labs, in particular
  - SQL injection
  - Cross-Site Scripting
  - Cross-Site Request Forgery
  - Races
  - Buffer overflow
- Optional lab on Real World Attacks
- Labs coordinated by Teaching Assistant Sam Thomas

SEED labs



http://www.cis.syr.edu/~wedu/seed/labs.html

Group Presentations

- Goal : deeper study of some secure programming aspect that is particularly relevant to you
  - New particular instance of a general bug class
  - Application of a general principle to a different language
  - New countermeasure tool or further exploration of a tool mentioned
- This is your continuous assessment (20% of overal mark)
- Evaluation : technical understanding (50%), novelty wrt lectures (25%), presentation (25%)

Organization

- Lecture hours :
  - Tuesdays 2pm (always)
  - Most Fridays 9am (see next slide)
- Office hours :
  - Friday 4pm-6pm if morning lecture (see next slide)
  - Thursday 4pm-6pm otherwise
- Labs :
  - Monday 12pm-1pm
  - Monday 12pm-2pm in Weeks 3,6,9
- Presentations
  - Choose your topic and group by end of Week 3
  - Report due end of Week 8 ; slides due end of Week 9
  - Presentations in Week 10

Tentative Schedule

week	Monday 12pm	Monday 1pm	Tuesday 2pm	Friday 9am	Assignments
1			1. Introduction	2. General Principles	
2	Setting up lab		3. Code injection (intro)	3. Code Injection (SQL)	
3	SQL injection lab	SQL Injection lab	3. Code Injection (XSS)		Choice topic and group for presentations
4	XSS lab		4. http sessions	4. http sessions	Formative assignment received
5	XSS lab		5. Unix	5.Unix	
6	CSRF lab	CSRF lab	3. Code injection (command)		Formative assignment returned
7	Race lab		6.Races	Feedback Formative assignment	
8	Race lab		7. Overflows (integer)	7. Overflows (buffer)	Send group report
9	Buffer overflow lab	Buffer overflow lab	8. Code Review		Send presentation slides
10	Student presentations		Student presentations	Student presentations	
11	Real World Attacks lab (optional)	Real World Attacks lab (optional)	Real World Attacks lab (optional)	Real World Attacks lab (optional)	

Lecture hours	**Lab hours**	Information related to assignments
- Tuesday 2pm Mech G34 - Friday 9am SPTX-LT1	Monday 12pm CS labs	

Office hours

Week	Day	Secure Programming	Other matters
1	Fri	3 :30 - 5 :30pm	2 :30 - 3 :30pm
2	Fri	3 :30 - 5 :30pm	2 :30 - 3 :30pm
3	Thu	3 :30 - 5 :30pm	2 :30 - 3 :30pm
4	Fri	3 :30 - 5 :30pm	2 :30 - 3 :30pm
5	Fri	3 :30 - 5 :30pm	2 :30 - 3 :30pm
6	Thu	3 :30 - 5 :30pm	2 :30 - 3 :30pm
7	Fri	3 :30 - 5 :30pm	2 :30 - 3 :30pm
8	Fri	3 :30 - 5 :30pm	2 :30 - 3 :30pm
9	Thu	3 :30 - 5 :30pm	2 :30 - 3 :30pm
10	Fri	3 :30 - 5 :30pm	2 :30 - 3 :30pm
11	Fri	3 :30 - 5 :30pm	2 :30 - 3 :30pm

Canvas

- Canvas page has all relevant information on the module
- Will be regularly updated, so check it !
- I aim to upload a draft version of lecture slides a week before the lecture
- May be updated slightly after lecture
- Please report any typo and error !

Books

- David D Wheeler, *Secure Programming for Linux and Unix HOWTO - Creating Secure Software*
- Howard and LeBlanc, *Writing Secure Code*
- Viega and McGraw, *Building Secure Software*
- Brian Chess, Jacob West, *Secure Programming with Static Analysis*
- Dieter Gollmann *Computer Security*

Websites

- CWE `cwe.mitre.org/`
- OWASP `www.owasp.org/`
- `www.owasp.org/index.php/How_to_write_insecure_code`
- Bugtraq mailing list `seclists.org/bugtraq/`
- CERT : `www.cert.org/`
- Common Criteria `www.commoncriteriaportal.org`

Secure Programming Training

- SEED labs `www.cis.syr.edu/~wedu/seed/labs.html`
- OWASP WebGoat project `www.owasp.org/index.php/Category:OWASP_WebGoat_Project`
- `security.cs.rpi.edu/courses/binexp-spring2015/`

Feedback is very welcome

- Do not wait end of term to tell me if you are lost !
- I am keen to improve this course
  - Slides or explanations unclear
  - Typos and errors
  - Content that might be added/removed
- Best moments for feedback are
  - Just after lecture hours
  - During office hours
  - By email anytime

Acknowledgements

- While preparing this course I used teaching material developed by Erik Tew at the University of Birmingham (kindly provided to me) and Meelis Roos at Tartu University (available on the web)
- Some of my slides are heavily inspired from theirs (but blame me for any errors !)