Secure Software Design

Mario Alviano

University of Calabria, Italy

A.Y. 2019/2020

Outline

1 Organization

2 Introduction
- Security thinking
- Security notions
- Controlling a computer

3 Overview of the course

About me

- Mario Alviano
  - First and second degrees in Computer Science
  - PhD in Computer Science — Logic programming for AI
  - For details: `http://www.alviano.net/`
- Consultation hour
  - Tuesday 16:00 – 17:00
  - Check my website for changes
  - You may write me an e-mail to check if I will be in my office

Course web page

```
https://www.mat.unical.it/ComputerScience/
SecureSoftwareDesign
```

Hint

- You can receive update messages via email
  1. Register yourself on the wiki
     (unless you already did)
  2. Subscribe on the page

Schedule

When?

Wednesday	10:30 – 13:30
Thursday	08:30 – 10:30

What?

Lectures and exercises, including PC exercises

Where?

MT15

Check the web page for possible changes!

Exams and attendance

Exams

- Written, including PC exercises
- Dates to be fixed
- Homeworks presented in the class matter!
  (Up to around 3 bonus points on the first exam after the course)

Attendance

- Attendance of the lectures is mandatory
- To access the exam you have to attend at least 70% of the course

Teaching material

Slides and material on the web page

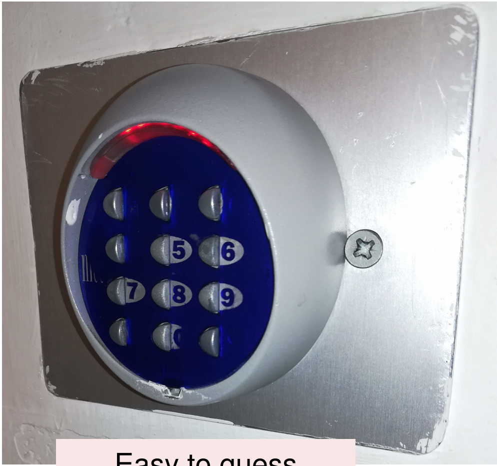
`https://www.mat.unical.it/ComputerScience/SecureSoftwareDesign`

Suggested books

1. Allen Harper et al.
   Gray Hat Hacking: The Ethical Hacker's Handbook
2. The CERT Oracle Secure Coding Standard for Java
3. Richard E. Smith
   Elementary Information Security
4. Chuck Easttom
   System Forensics, Investigation & Response

Any problem here?

Difficult to shield



Easy to guess

Security decision-making

Three categories:

1. Rule-based decisions
   - Established, widely accepted guidelines
   - Example: car ignition lock
2. Relativistic decisions
   - Outdo others
   - Example: hunter’s dilemma
3. Requirements-based decisions
   - Systematic analysis of the security situation
   - Example: Risk Management Framework

Risk Management Framework (1)

1. Categorize the information system
2. Select security controls
3. Implement security controls
4. Assess security controls
5. Authorize the information system
6. Monitor security

Continuous Improvement

The process never ends at the final step.

Risk Management Framework (2)

Security Category

High-level estimate of the impact of failures

The CIA properties:

- Confidentiality
- Integrity
- Availability

Potential impact for each property:

- Not applicable (NA; only for confidentiality)
- Low impact
- Moderate impact
- High impact

SC *name* = {(confidentiality, *impact*), (integrity, *impact*), (availability, *impact*)}

Security boundaries

- The essence of any protection
- Establish a container for our assets
- Protect assets by denying access to threat agents
- Degree of protection in terms of strength of the boundary

Least Priviledge

- Limit the number of people allowed inside the security boundary
- If possible, restrict what each person may do to the asset

Security architecture

- Decompose the system into separate security domains
- Each domain has its own security boundaries
- Security domains may have a hierarchical structure

Defence in Depth (or layered defence)

Separate security domains shall provide separate layers of protection

Threat agents

- Who threatens our assets?
- Individuals are not important
- We are interested in identifying categories of people
- Those are our threat agents

Potential attacks

What attacks arise when CIA properties fail?

- Disclosure
- Subversion
  - Forgery
  - Masquerade
- Denial of service (DOS)

Ethical issues

When an organization requests a security assessment

- The analyst needs written authorization
- The analyst should use the appropriate tools
- When finished, the analyst should report to the appropriate people in the organization

Responsible disclosure

- The finder reports the vulnerability to the vendor
- The vendor acknowledge the report within 7 days
- The vendor provides weekly updates to the finder
- The vendor and the finder should jointly decide how to announce the vulnerability
- If no agreement, the finder will provide a general announcement 30 days after the vendor was informed
- Announcements should **not** include details that allow an attacker to exploit the vulnerability

Program execution

- The CPU executes machine instructions
- A register stores the program counter (PC)
- Conditional instructions are used to break sequentiality
- The control section of the memory stores instructions
- The data section contains program's data

Such a separation is not always checked

Procedures

- Programs are often split in procedures
- Calling a procedure requires to modify PC
- After the called procedure terminates, PC must return to the callee procedure
- This is achieved by storing the return address in the stack
- Buffers local to a procedure are also stored in the stack

A buffer overflow may replace the return address

The Morris Worm

The `finger` program

Provides information on user@machine

It had a buffer overflow

- Assumed that people would rely on short names
- Allocated only 11 bytes for user@machine (plus null character)
- Morris provided a long string to execute a shellcode

Why this attack was possible

- Separation of data and instructions was not checked
- `finger` ran with root privilege

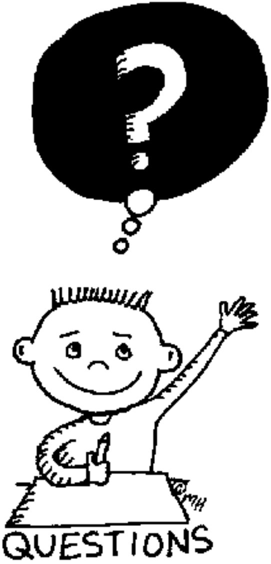
Security alerts

Computer Emergency Response Team (CERT)

- An official clearinghouse for reporting vulnerabilities
- Published CERT Advisories for many years
- CERT Advisory numbers are used to refer well-known vulnerabilities
- Today, we also use CVE numbers, from the Common Vulnerability Enumeration database

Overview of the course

- Most frequent weaknesses in coding
- Noncompliant and compliant code examples
- Exploit exercises
- Assembly and low level attacks



END OF THE LECTURE