Secure Programming

COMPUTER SYSTEMS SECURITY

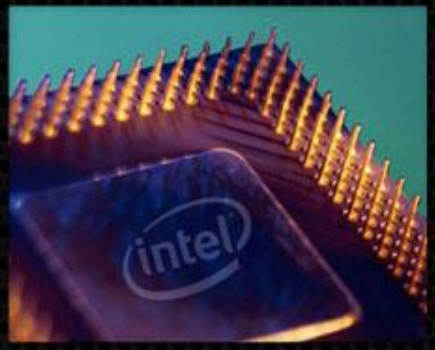
CARLOS MIGUEL CORREIA DA COSTA
VÍTOR AMÁLIO MAIA MARTINS MOREIRA

Context

- Our world is increasingly more computerized
  - Software is used in nearly all professional activities
- Software attacks can cause loss of data or money
  - Some can compromise the safety of people
- Cyber espionage and cyber warfare
  - Threatens data privacy and systems security worldwide

LIFE-CRITICAL SYSTEM VERIFICATION

"If it fails, people die."



Theoretical computer scientists harness the power of logic and mathematics to provide a provable guarantee of safety.

Goals

- Research software vulnerabilities
  - How can we avoid most common exploits
- Research secure programming methodologies
  - How can we approach secure system implementations
- What tools can help in building secure systems
  - Static analyzers
  - Exploits frameworks

Software vulnerabilities

- Memory safety
  - Change programs variables to change its behavior
    - Buffer overflows
    - Heap spraying
- Denial of Service
  - Affect the availability of a system
    - DoS
    - DDoS

Software vulnerabilities

- Privilege escalation
  - Gain access to protected resources
    - Time of check to time of use
    - Symlink race
- Browser exploits
  - Gain access to other webpages data or bypass authentication protocols
    - Cross-site scripting
    - Cross-site tracing
    - Cross-site request forgery

Software vulnerabilities

- Improper input validation
  - Gain unauthorized access to databases
    - SQL injections
  - Code injections
  - Remote file inclusion
- Network protocols
  - Phishing
  - DNS and email spoofing

Hardware vulnerabilities

- Cold boot attack
  - Access encryptions keys resident in RAM
- Smudge attack
  - Detect login information in touch screen

Common programming errors

- Failing to check return value
- Bad use of software libraries or functions
- Failing to sanitize input
- Failing to check memory bounds

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
        char buffer[20];

        strcpy(buffer, argv[0]);
        printf("%s", buffer);

        return 0;
}
```

Other programming errors

- Using compromised 3rd party components
- Bad architectures (sometimes it's not the software)

Secure programming methodologies

- Coding standards
- Norms used to write good software
- Choosing good architectures
- And of course, tools!

Secure programming tools

- Static code analyzers
- Vulnerability testing tools
- Sandboxes
- Virtual machines

Questions