Introduction to Software Security
Coursework 01

Introduction

This coursework introduces students to basic approaches to specify and verify security aspects concerning *confidentiality*, *integrity* and *availability* for software systems. In particular, this coursework provides theoretical and practical exercises to *(i)* identify and describe software vulnerabilities concerning memory safety in C programs; *(ii)* describe and evaluate typical examples of SQL injection, which can be exploited by an attacker; *(iii)* specify and verify race conditions in concurrent software; *(iv)* identity and describe cyber-threats in a simple embedded system, which can be remotely operated; and lastly *(v)* describe how (bounded) model checking techniques work.

Learning Objectives

By the end of this lab you will be able to:

- Define standard notions of security and use them to evaluate the system's confidentiality, integrity and availability.
- Explain standard software security problems in real-world applications.
- Use testing and verification techniques to reason about the system's safety and security.

1) **(Critical Software Vulnerabilities)** Identify and describe the critical software vulnerabilities from the program statements a) to e) considering the following C code.

```
int *zPtr;
int *aPtr = NULL;
void *sPtr = NULL;
int number, i;
int z[5] = {1, 2, 3, 4, 5};
sPtr = z;
```

a) `++zPtr;`
b) `number = zPtr;`
c) `number = *zPtr[2];`
d) `number = *sPtr;`
e) `++z;`

2) **(SQL injection)** SQL injection allows an attacker to interfere with the queries to the database in order to retrieve data. For example, a programmer can construct a SQL query to check name and password as

```
query = "select * from users where
name='" + name + "'" and pw = '" +
password + "'"
```

Figure 2: SQL Injection Example.

If an attacker provides the name string, the attacker can set name to "John' –"; this would remove the password check from the query (note that -- starts a comment in SQL). Provide here other examples of SQL injection:

i. Retrieving hidden data;
ii. Subverting application logic;
iii. UNION attacks;
iv. Examining the database;
v. Blind SQL injection.

3) **(Race Condition)** Specify and verify the three properties as specified below for the following mutual exclusion algorithm:

int flag[2], turn, x, i;

void *t1(void *arg) {
flag[0] = 1;
turn = 1;
while (flag[1] == 1 && turn == 1) {};
//critical section
if (i==1) x=1;
//end of critical section
flag[0] = 0;
return NULL;
}
	void *t2(void *arg) {
	flag[1] = 1;
	turn = 0;
	while (flag[0] == 1 && turn == 0) {};
	//critical section
	if (i==2) x=2;
	//end of critical section
	flag[1] = 0;
	return NULL;
	}

i. At most, one process in the critical region at any time.
ii. Whenever a process tries to enter its critical region, it will eventually succeed.
iii. A process can eventually ask to enter its critical region.

4) **(Cyber-threats)** A typical embedded system consists of human-machine interface (keyboard and display), processing unit (real-time computer system), and instrumentation interface (sensor, network, and actuator) that can be connected to some physical plant. In particular, the overall objective of the simple embedded system illustrated in Figure 1 is to keep the temperature and pressure of some chemical process within well-defined limits by a remote operator.

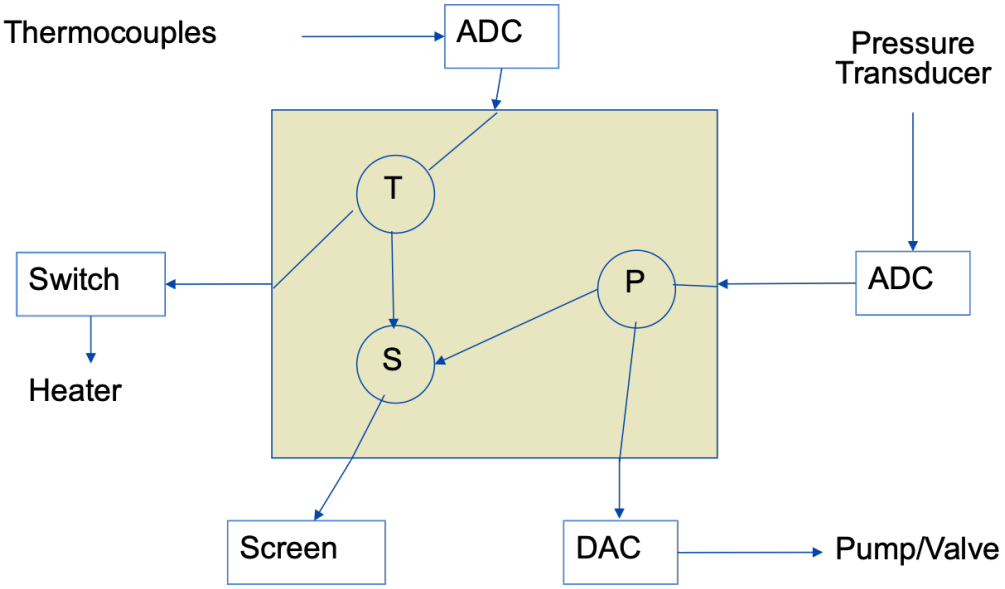

Figure 1: Simple Embedded System.

i. Specify the following properties of this simple embedded system:

a) A **T** process reads the measured values from a temperature sensor and turns the heating system on if the temperature is below 20°C and turns off the heating system if the temperature is above 300°C.
b) Process **P** regulates pressure with a sensor opening the pump/valve when the pressure value is above 500bar and closing the pump/valve when the pressure value is below 100bar.
c) Whenever the **T** and **P** processes transfer data to an **S** process, the measured values will be shown on a liquid crystal display (LCD).

ii. What are the **security objectives** of this simple embedded system?

iii. What are the sources of **security problems**?

5) **(Bounded Model Checking)** Describe the main steps involved in checking programs using the BMC technique, from reading the program to generating the SMT formulas. Consider the following example to describe each step in the technique.

```
int main(int argc, char **argv) {
  long long int i=1, sn=0;
  unsigned int n=5;
  assume (n>=1);
  while (i<=n) {
    sn = sn + a;
    i++;
  }
  assert(sn==n*a);
}
```