CS 458: Computer Security and Privacy

Michael Noukhovitch

Spring 2016, University of Waterloo

Notes written from Erinn Atawater's lectures.

Contents

1 Introduction

1.1 Security

Security can be defined as:

confidentiality access to systems is limited to authorized

integrity getting the correct data

availability system is there when you want it

1.2 Privacy

There are many definitions but we will stick to **informational self-determination**, where you control the information about you

1.3 Terminology

assets things we want to protect

vulnerabilities weaknesses in a system that can be exploited

threats loss or harm that may befall a system

- interception
- interruption
- modification
- fabrication

threat model set of threats to defend against (who/what)

attack an action which exploits a vulnerability to execute a threat

control removing or reducing a vulnerability

1.4 Types of Defence

Defend against an attack:

- **prevent** stop the attack from happening
- **deter** make the attack more difficult
- **deflect** make it less attractive for attacker
- **recover** mitigate effects of the attack

Make sure that defence is correct with principles:

- **easiest penetration** system is only as strong as weakest link
- **adequate protection** don't spend more on defence than the value of the system

1.5 Methods of Defence

- Software controls: passwords, virus scanner ...
- Hardware controls: fingerprint reader, smart token ...
- Physical controls: locks, guards, backups ...
- Policies: teaching employees, password changing rules

2 Program Security

2.1 Flaws, faults, and failures

2.1.1 Defintions

flaw problem with a program

fault a potential error inside the logic

failure an actual error visible by the user

2.1.2 Unexpected Behaviour

A spec will list the things a program will do but an implementation may have additional behaviour. This can cause issues as these behaviours might not be tested and would be hard to test.

2.2 Unintentional Security Flaws

2.2.1 Types of Flaws

- **intentional**
  - **malicious**: inserted to attack system
  - **nonmalicious**: intentional features meant to be in the system but can cause issues
- most flaws are **unintentional**

2.3 Buffer Overflow

2.3.1 Definition

Most common exploited type of security flaw when program reads or writes past the bounds of the memory that it should use. If the attacker exploits it they can override things like the *saved return address*. Targets programs on a local machine that run with setuid priveleges or network daemons

Example 2.1. basic buffer overflow

```
#define LINELEN 1024

char buffer[LINELEN];
strcpy(buffer, argv[1]);
```

2.3.2 Types

- only a single byte can be written past the end of the buffer
- overflow of buffers on the heap (instead of the stack)
- jump to other parts of the program or libraries (instead of shellcode)

2.3.3 Defences

- language with bounds-checking
- non-executable stack (mem is never both writable and executable)
- stack at random virtual addresses for each process
- "canaries" detect if stack has been overwritten before return

2.4 Integer Overflows

Program may assume integer is always positive, and below certain value. Overflow will make a too large singed integer negative, violating assumptions.

2.5 Format String Vulnerabilities

Format strings can have unexpected consequences:

- `printf(buffer)` parse buffer for \%s and use whatever is on the stack to process found format params
- `printf(%s%s%s%s)` may crash your program
- `printf(%x%x%x%x)` dumps parts of the stack
- `%n` will write to an address on the stack

2.6 Incomplete Mediation