CS-527 Software Security

Basic Security Principles

Asst. Prof. Mathias Payer

Department of Computer Science
Purdue University

TA: Kyriakos Ispoglou

`https://nebelwelt.net/teaching/17-527-SoftSec/`

Spring 2017

Table of Contents

Software Security

Definition

Allow intended use of software, prevent unintended use that may cause harm.

Security principles

Confidentiality: an attacker cannot recover protected data.
Integrity: an attacker cannot modify protected data.
Availability: an attacker cannot stop/hinder computation.

These fundamental **CIA** properties are used in different contexts of software security policies and mechanisms.

Accountability/non-repudiation is sometimes mentioned as fourth fundamental concept. It prevents denial of message transmission or receipt. Non-repudiation is, at its core, a legal concept.

Security analysis

Given a software system, is the system secure?

- ... it depends
- What is the interface? (What is the attack surface?)
- What are the assets? (How profitable is an attack?)
- What are the goals? (What drives an attacker?)

Threat model

Definition: Threat model

Threat model defines an attacker's abilities and resources.

Assume you want to protect an asset (your valuables) by locking them in a safe.

- In cloud cuckoo land you don't need to lock your safe.
- An attacker might pick your lock.
- An attacker might use a torch to open the safe.
- An attacker may have some advanced safe opening technology (e.g., an X-ray).
- An attacker may know you and get access to (or copy) your key.

Cost of security

There's no free lunch, security will incur overhead.
Security is...

- expensive to develop,
- may have performance overhead,
- may be inconvenient to users.

Table of Contents

Security principles

Least privilege

The principle of least privilege ensures that a component has the least privileges needed to function. Any privilege that is further removed from the component removes some functionality. Any additional privilege is not needed to run the component according to the specification. Note that this property constrains an attacker in the privileges that can be obtained.

Security goals

Separation

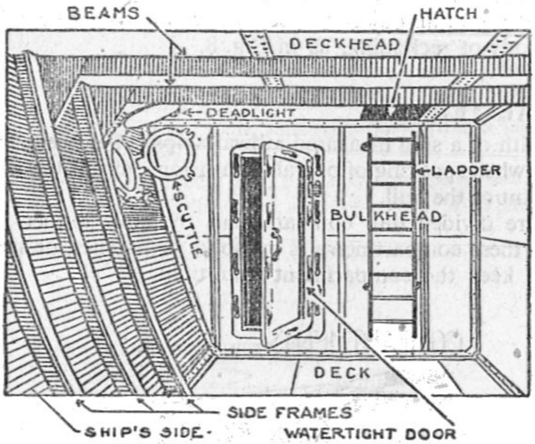
Separate individual components into smallest functional entity possible. General idea: contain faults to individual components. Allows abstraction and permission checks at boundaries. Note that this property builds on least privilege. Both properties are most effective in combination: many small components that are running and interacting with least privileges.

Security goals

Trust and correctness

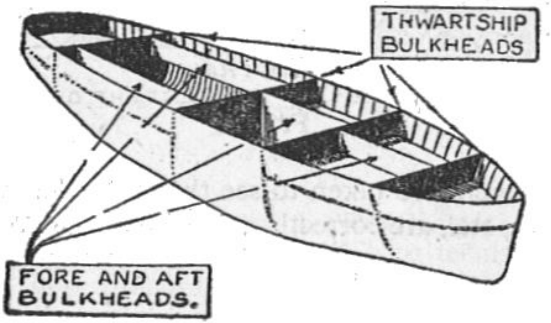
Specific components are assumed to be trusted or correct according to a specification. Formal verification ensures that a component correctly implements a given specification and can therefore be trusted. Note that this property is an ideal property that cannot generally be achieved.

Security scope



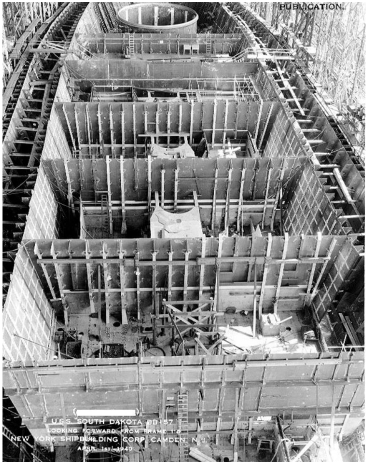
(Parts of a water-tight compartment, Seaman's Pocket Book, 1943)

Security scope



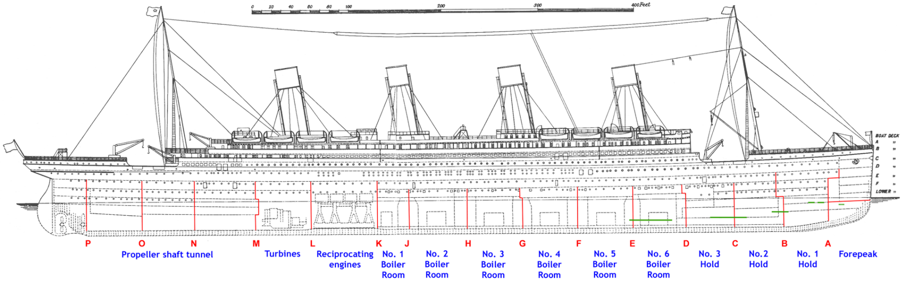
(Parts of a water-tight compartment, Seaman's Pocket Book, 1943)

Security scope


PUBLICATION.
U.S.S. SOUTH DAKOTA BB-57
LOOKING FORWARD FROM FRAME 116
NEW YORK SHIPBUILDING CORP. CAMDEN, N.J.
APRIL 1st 1940

(USS South Dakota (BB-57) at the New York Shipbuilding Corporation shipyard, Camden, New Jersey, 1 April 1940.)

Security scope



(Side plan of the RMS Titanic, 1911, annotated)

Table of Contents

Operating System (OS) abstraction

- Allows individual processes exclusive access to resources.
- Provides well-defined API to access hardware resources.
- Enforces mutual exclusion to resources.
- Enforces access permissions for resources.
- Restricts attacker to specific user privileges.
- ... is a trusted component. All process can be attacked through vulnerabilities in the OS kernel.
- ... trusts the hardware.

Hardware abstraction

- Virtual memory is a hardware-enforced separation of memory pages to individual processes.
- Only operating system has access to physical memory (or at least believes so).
- DMA allows trusted devices direct access to memory.
- ISA implements different privilege level (ring 0 to ring 3 on x86).
- Hardware abstractions are fundamental for performance.

Table of Contents

Access control

Access control specifies how entities interact with resources.

Authentication: Who are you (something you know, have, or are)?

Authorization: Who has access to object? (Follows authentication)

Audit: I'll check what you did.

Types of access control

Discretionary Access Control (DAC): Object owners specify policy.

Mandatory Access Control (MAC): Rule and lattice-based policy.

Role-Based Access Control (RBAC): Policy defined in terms of roles (sets of permissions), individuals are assigned roles, roles are authorized for tasks.

Different security models

- Access control lists
- Capabilities
- Bell - LaPadula
- Information flow

Access control matrix

Goal: provide access rights for subjects to corresponding objects[1].

	foo	bar	baz
user	rwx	rw	rw
group	rx	r	r
other	rx		r

Used, e.g., in the Unix/Linux security model for file accesses or the Android, iOS, or Java security model for privileged APIs.

[1]Introduced by Butler Lampson in 1971, `http://research.microsoft.com/en-us/um/people/blampson/08-Protection/Acrobat.pdf`.

Table of Contents

Summary

- Software security goal: allow intended use of software, prevent unintended use that may cause harm.
- Three principles: Confidentiality, Integrity, Availability.
- Security of a system depends on its threat model.
- Least privilege, separation, and trust as concepts.
- Security relies on abstractions to reduce complexity.
- Reading assignment: Butler Lampson, Protection, `http://doi.acm.org/10.1145/775265.775268`.

Questions?

?