DM510: Security and Protection

Lars Rohwedder

Disclaimer

These slides contain (modified) content and media from the official Operating System Concepts slides: https://www.os-book.com/OS10/slide-dir/index.html

Today's lecture

- Chapters 14+15 of course book

Security Overview

Security violations

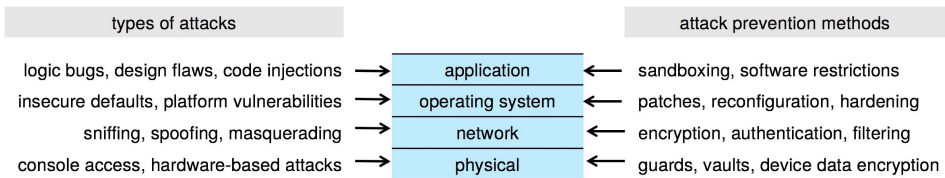
- **Breach of confidentiality:** Unauthorized reading of data
- **Breach of integrity:** Unauthorized modification of data
- **Breach of availability:** Unauthorized destruction of data
- **Theft of service:** Unauthorized use of resources
- **Denial of service (DOS):** Prevention of legitimate use

Security measures

- Absolute security impossible $\rightsquigarrow$ make cost to perpetrator sufficiently high
- Secure all four layers: security as weak as weakest link of the chain

Four layers of security

- **Physical:** Data centers, servers, connected terminals
- **Application:** Benign or malicious apps can cause security problems
- **Operating System:** Protection mechanisms, debugging
- **Network:** Intercepted communications, interruption, DOS



Application threats

Malicious programs

Typical goals of malicious programs:

- **Spyware:** display adds, sniff sensitive data (passwords, etc.)
- **Ransomware:** encrypt data to get ransom
- Gain control of system for distributed DOS attacks, bitcoin mining ...

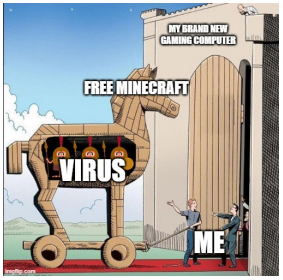
Malicious programs

Typical goals of malicious programs:

- **Spyware:** display adds, sniff sensitive data (passwords, etc.)
- **Ransomware:** encrypt data to get ransom
- Gain control of system for distributed DOS attacks, bitcoin mining ...

Types of malicious programs:

- **Trojan horse:** malicious code is part of seemingly useful program
- **Virus/worm:** malicious code self-replicates by attaching itself to other programs, sometimes across networks

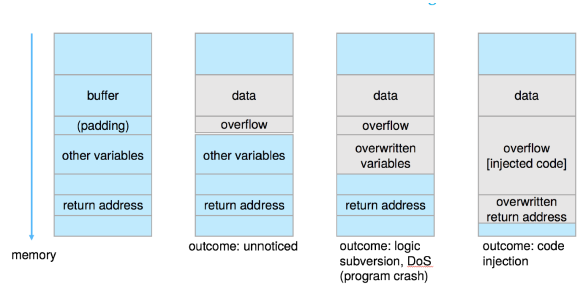


Code injections

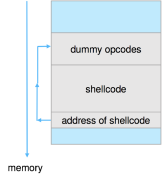
- Program is not malicious, but attackers can make it execute their code

Buffer overflow

- Program does not check buffer limits and writes beyond it
- Can lead to input being written over other data or even return addresses



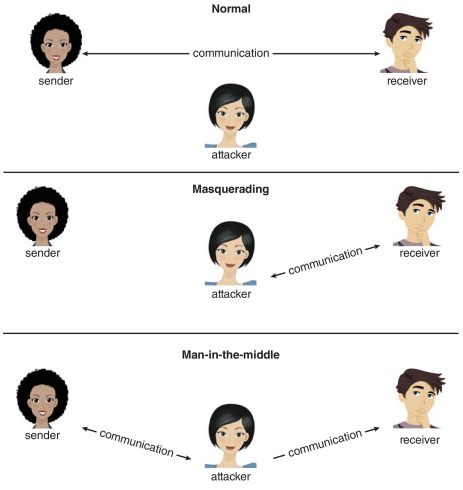
- Attackers may redirect return address to memory segment they control - to execute arbitrary code
- **Prevention:** better code practices, safer languages or checks inserted by compiler, bits in page table for non-executable data



Network threats

Types of network attacks

- We assume we cannot trust network channels
- Attacker may listen to communication
- **Masquerading:** attacker pretends to be someone else
- **Man-in-the-middle:** attacker pretends to be opposite endpoint to sender and receiver and forwards (modified) communication between them
- Main tool for prevention: **Cryptography**



Cryptography

Goal: authentificate (confirm correct identity) or ensure plaintext is received only by correct destination

Given

- **Message set M:** all possible plaintexts, for example, blocks of 1024 bytes
- **Cypher set C:** all possible encrypted messages, for example, blocks of 1024 bytes
- **Keys K:** data used to encrypt/decrypt

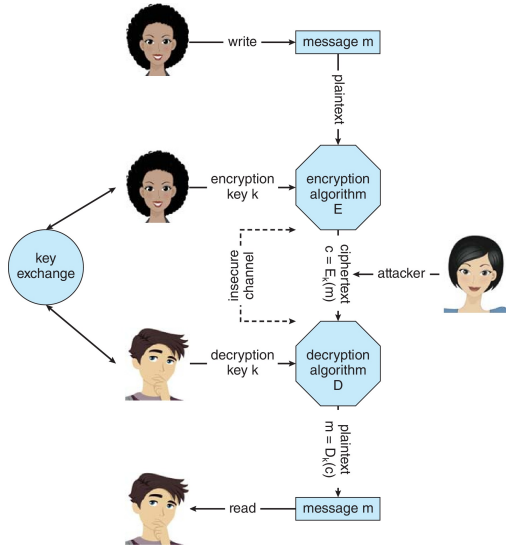
Basic functions

- **Encryption:** for each key $k \in K$, we obtain encryption function $E_k : M \to C$
- **Decryption:** for each key $k \in K$, we obtain decryption function $D_k : C \to M$

Crucial assumption: $E_k(m)$ and $D_k(c)$ can only be computed if k known.

Symmetric encryption

- Same key is used for encryption and decryption: $D_k(E_k(m)) = m$
- Sender encrypts and transmits cypher, receiver decrypts
- Only cipher is transmitted, so attacked cannot recover message
- **Applications:** hard drive encryption, end-to-end encryption across networks
- But how to safely perform key exchange?



Asymmetric encryption

- Pair of keys: **public key** and **private key**. One to encrypt, other to decrypt
- Relies on advanced algorithms, for example RSA

Authentification

Alice wants to sign message m proving that she is origin

- Alice has private key k_1
- Public key k_2 is known by everyone
- $D_{k_2}(E_{k_1}(m)) = m$
- Only Alice is able to construct the cypher $c = E_{k_1}(m)$ and everyone can verify it by using the public key

Receipt only by certain destination

Alice wants to send message to Bob that only Bob can read

- Bob has private key k_1
- Public key k_2 is known by everyone
- $D_{k_1}(E_{k_2}(m)) = m$
- Alice sends cypher $c = E_{k_2}(m)$ and only Bob can decrypt it

Hash functions

- Function $H : M \to C$ (like encryption, but not meant to be decrypted)
- Usually hashed values C much smaller than M
- Should be hard to invert, i.e., for $c \in C$ finding some $m \in M$ with $H(m) = c$

Signatures of hashes

- Instead of encrypting entire message for authentification, encrypt only hash of message
- Others can also compute hash and check it against decrypted cypher
- More efficient (fewer bytes to encrypt/decrypt) and message can be transmitted in plaintext

Hashed passwords

- Storing or transmitting plaintext passwords should be avoided
- Safer: use hashed password
- Even safer: use hash of password with added **salt**, i.e., add random salt bytes to password (stored with hash). Prevents attacker from trying through list of known hashes

Protection

Goals of protection

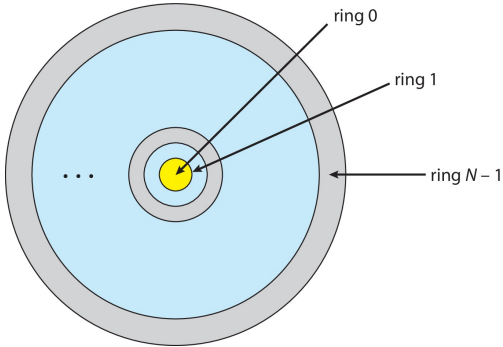
- **Protection problem (security measures by operating system):** Ensure that each object (hardware or software) is accessed correctly and only by those processes that are allowed to do so
- **Mechanism:** how something should be done
- **Policy:** what should be done
- Mechanism and policy should be separated for flexibility

Principles

- **Principle of least priviledge:** give each process and user only enough priviledges to complete its task
- **Need-to-know principle:** only the information required for its task should be shared with process (similar to principle of least priviledge)
- **Compartmentalization:** protect each system component by permissions and access restrictions

Protection rings

- Hierachical abstraction for different priviledge levels
- Process runs at specific ring level
- Innermost ring corresponds to having all priviledges
- Each ring has subset of inner ring's priviledges
- Often requires hardware support, for example, user mode and kernel mode

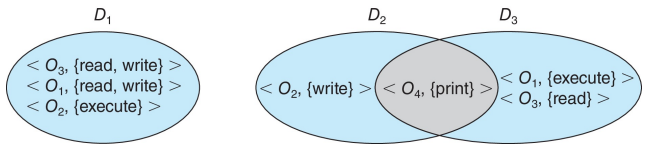


Protection domains

- Processes operate with priviledges of specific domain
- Domain has rights over various operations for each object
- In **dynamic** setting, processes can switch domains
- More fine-grained than hierachical protection rings

In Unix

- Domain is user-id
- Can change user-id via `su`, but may require password



Access matrix

- A matrix allows specifying arbitrary policies
- Storing the entire matrix explicitly usually not efficient, since it is very sparse. Alternatives: capability-list for objects (list of domains and allowed operations) or capability-lists for domains

domain \ object	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			