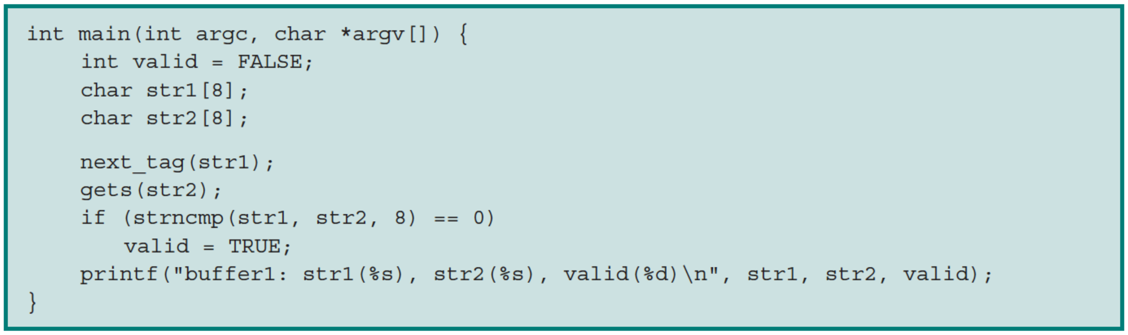
Chapter 15

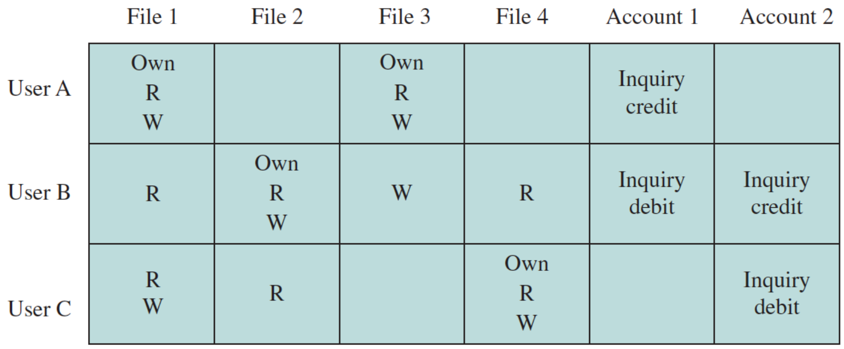
- **Confidentiality** – ensuring access to information is only given to authorized entities
- **Integrity** – ensure that information is consistent, accurate
- **Availability** – ensure that a service is available in the future when needed
- Categories of system access threats:
    - **Intruders**
        - Masquerader
        - Misfeasor
        - Clandestine user
    - **Malicious software**
        - Parasitic vs. self-contained, independent
        - Non-replicating, activated by trigger vs. replicating
- Countermeasures:
    - **Intrusion detection systems (IDS)**
        - Host-based vs. network-based
        - Logical components:
            - Sensors
            - Analyzers
            - User interface
    - **Authentication** – process of verifying an identify claimed by/for a system entity
        - <u>Identification</u> – user provides a claimed identify to the system
        - <u>Verification</u> – establishing the validity of that claim
        - Means of authentication:
            - Something the individual *knows*
            - Something the individual *possesses*
            - Something the individual *is* (static biometrics)
            - Something the individual *does* (dynamic biometrics)
    - **Access control** – specifies which user/process can have access (and the type of access) to each system resource
        - Access control database keeps track of the type of access given to each user, for each resource
    - **Firewall** – protects a system from <u>external</u> (network-based) threats
        - Design goals:
            - Acts as choke point – all traffic pass through firewall
            - Enforces local security policy
            - Secure against attacks
- **Buffer overflow** – condition under which more input than the allocated capacity can be placed into a buffer, thereby overwriting other information
    - E.g. inputting string >8 characters will fill past the capacity of str2 and overwrite the contents of str1



```
int main(int argc, char *argv[]) {
    int valid = FALSE;
    char str1[8];
    char str2[8];

    next_tag(str1);
    gets(str2);
    if (strncmp(str1, str2, 8) == 0)
       valid = TRUE;
    printf("buffer1: str1(%s), str2(%s), valid(%d)\n", str1, str2, valid);
}
```

  - Compile-time defenses:
    - Programming language (static typing, array bounds checking)
    - Safe coding techniques (auditing codebases, careful pointer handling)
    - Language extensions & safe libraries
    - Stack protection mechanisms (verify canary value is not modified)
  - Run-time defenses
    - Address space protection – make the stack & heap non-executable
    - Address space randomization – randomize location of process stacks
    - Guard pages – placed between critical regions of memory
- File system access control
  - **Access matrix**
    - Subject – accessing user/process
    - Object – accessed file, memory, etc.
    - Access right – read, write, execute, etc.



	File 1	File 2	File 3	File 4	Account 1	Account 2
User A	Own R W		Own R W		Inquiry credit	
User B	R	Own R W	W	R	Inquiry debit	Inquiry credit
User C	R W	R		Own R W		Inquiry debit

    - Decompose by columns → access control lists for each file
    - Decompose by rows → capability tickets for each user
  - Access control policies
    - Discretionary (DAC) – access right can be passed to another entity
    - Mandatory (MAC) – access right cannot be passed on
    - Role-based (RBAC) – access based on rules for roles
- UNIX file access control
  - Each file has 12 protection bits
    - 9 permission bits – read/write/execute for owner, group, others
    - Set user/group ID bits (2 bits)
    - Sticky bit

- Hardening an operating system:
    - Install & patch the OS
    - Harden & configure the OS by:
        - Removing unnecessary service/applications/protocols
        - Configuring users, groups, permissions
        - Configuring resource controls
    - Install & configure additional security controls (e.g. antivirus, IDSs)
    - Test the security
- Security maintenance
    - Monitoring & analyzing logs
    - Performing regular backups
    - Recovering from security compromises
    - Regularly testing system security
    - Keeping all critical software patched & updated