Denial of Service (DoS)

An action that prevents or impairs the authorized use of networks, systems, or applications by exhausting resources such as CPU, memory, bandwidth, or disk space

Often uses source address spoofing

Examples:

DDoS, Reflector Attacks, Amplifier Attacks:

Defenses against DoS attacks:

- DoS attack prevention
- DoS attack response
Intrusion detection

The number of potential attacks has increased tremendously

Therefore, it is harder than ever to prevent attacks

Therefore, there is much more focus than ever before on attack detection and incident response

- Intrusion Detection System (IDS)
Underlying premise of IDS:

Computer systems that are not under attack exhibit the following characteristics:

Systems under attack fail to meet at least one of these characteristics

Overall goals of IDS

1.

2.

3.

4.
Models of Intrusion

Model classifies a sequence of states or actions as “good” (no intrusion) or “bad” (possible intrusion). There are 3 basic models used in IDS: anomaly; misuse; and specification

Anomaly model
- Statistical characterization
Misuse model
- Known (or believed) bad actions or states

Specification model
- Good program specifications

Combination of models
IDS Architecture

Typically, 3 components
- Agent(s)
- Director
- Notifier
- Agent:

- Director:

- Notifier:
Intrusion Response

Once an intrusion is detected, need to protect the system in a way that will minimize damage.

Want to restore the system to comply with the security policy and take any required actions against the attacker. The following steps are involved.

1. Preparation:

2. Identification of attack:

3. Containment:

4. Eradication:

5. Recovery:

6. Follow-up:
Firewalls

An additional layer of defense, insulating the internal systems from external networks

Goals:

May provide different types of control:

Capabilities and Limitations:

Four main types of firewalls:

Firewall installation, location, and configuration:

Intrusion Prevention Systems, Unified Threat Management:
Buffer overflow

The most pervasive security problem today in terms of reported bugs

The problem exists because it is so easy to write beyond the bounds of data objects in languages like C and C++ (but not Java and C#)

System’s loader puts data in 3 areas
- Data segment
- Stack
- Heap

Buffer overflow can work with stack-allocate buffers and with heap-allocated buffers

Very simple example:

```c
void main() {
    char buf[1024];
    gets(buf);
}
```

This gets (unbounded) input and uses it to load a stack variable.
Slightly more detailed example:

Several techniques are available to defend against buffer overflow attacks (compile-time defenses and run-time defenses)