جَـامعة
المَـنارة
MANARA UNIVERSITY

Information Systems Security

Fall 2025

Information Systems Security
Overview

جَـامعة
المَـنارة
MANARA UNIVERSITY

FIPS 200 requirements (1)

FIPS 200 (government document) defines high level security requirements

- **Access control**: Limit who gets in and what they can do
- **Awareness and training**: Prevent uninformed users aiding attack
- **Auditing and accountability**: Track who's doing what
- **Certification and assessment**: Periodically review security posture
- **Config management**: Track how things are configured, note changes
- **Contingency management**: Have plans for emergencies
- **Identification/authorization**: Check user identities
- **Incident response**: Plan how to respond during/after a breach
- **Maintenance**: Actively maintain systems (no deploy & forget!)

Note: This is hyper-summarized, of course

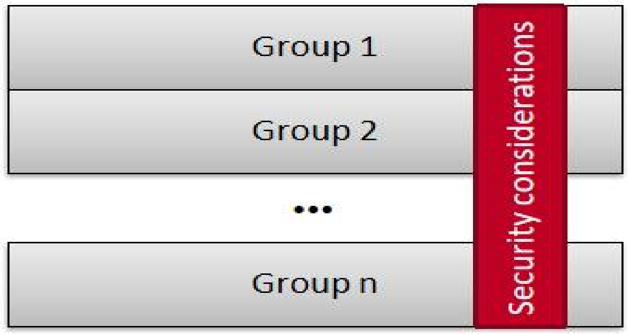
FIPS 200 requirements (2)

FIPS 200 (government document) defines high level security requirements

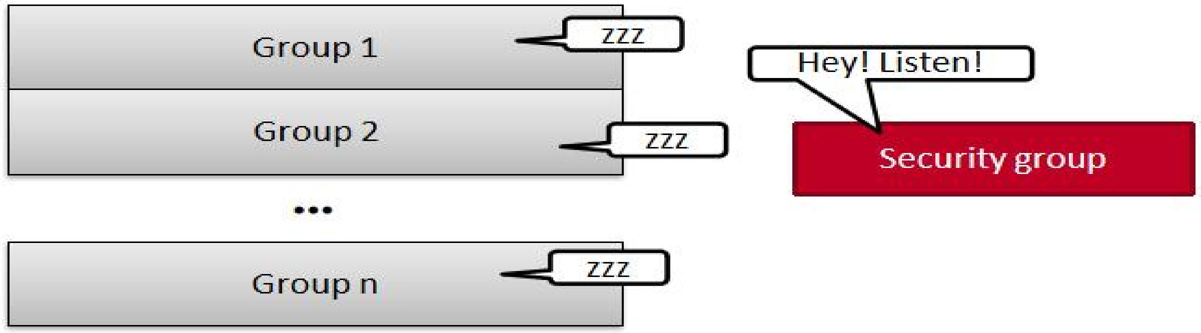
- **Media protection**: Keep storage safe (even when trashing it)
- **Physical/environmental protection**: Doors, walls, cameras, etc.
- **Planning**: Every action is planned then executed, no ‘cowboy IT’
- **Personnel security**: Vet the people working there
- **Risk assessment**: Analyze risk and invest proportionally
- **System and services acquisition**: Source goods/services wisely
- **System and communication protection**: Good software engineering
- **System and information integrity**: Malware countermeasures

Human and technology factors are *interwoven*

- Good model of security: “a thread through everything”

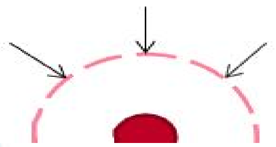


- Bad model of security: “a separate silo”



Attack surface

- For a system already deployed, you may want to assess risk
- One thing to ask: how many ways could an attacker interact with it?
- This is the **attack surface.**
  - Includes the software itself, the network, and humans.
  - Examples of attack surfaces in desktop operating systems:
    - Big attack surface: **Windows 95**, when it comes online, listens for connections on several port numbers with various large and complex services.
    - Smaller attack surface: **Windows 10**, when it comes online, listens on a few ports, and has a firewall that blocks most connections (but the firewall has exceptions by default that still allow some fairly complex services to listen).
    - Even smaller attack surface: **Ubuntu Linux 22.04**, when it comes online, listens on no ports whatsoever.
- **Good practice: find ways to reduce attack surface!**



Security strategy

1. **Specification/policy**: What is your goal? Consider tradeoffs against ease of use and cost.

2. **Implementation**: Identify mechanisms of <u>prevention</u>, <u>detection</u>, <u>response</u>, and <u>recovery</u>.

3. **Evaluation**: Don't assume it worked; prove it.

- The above seems simple, but ***I have seen one of these steps skipped SO MANY TIMES.***
  - I've seen people forget #1 (deploy and evaluate tools without regard for their needs)
  - I've seen people forget #2 (decide on goals, not fund the implementation, then get mad when they're not met)
  - I've seen people forget #3 (set up fire-and-forget security solutions that quietly die soon after)

Threat models

- When designing a defense, you must know the goal
- Define:
  - **Asset(s)** at risk
  - Type of **vulnerability** you assume exists and are protecting against
  - **Attacker's capabilities/knowledge**
- Only then can you say how your defense prevents the attack from succeeding despite the vulnerability (or detects it, response to it, or recovers from it).

Threat modeling example: HTTPS

HTTPS: Encrypted form of HTTP for secure web traffic

Threat model:

- **Asset(s)**: Private user communications, including credentials
- **Vulnerability**: Packets may be intercepted in transit (e.g. on open wifi)
- **Attacker's capabilities/knowledge**: Knows when/how to intercept packets for a specific user or for the site as a whole

The defense:

- **Our solution**: we negotiate a key in open communication known only to user and server; all content is encrypted with this key.
- **How it solves it**: Even with the full traffic, attacker cannot deduce key and therefore cannot decrypt communications. However, they do know that communication happened and roughly how much...

Why threat model?

- Threat models help us move from nebulous world of “more secure” to a specific guarantee

- MOST IMPORTANT: Promotes **systematic thinking** about when a defense can and cannot do

- Lets us compare techniques in terms of cost/benefit tradeoffs

- Understand what attacks are still on the table

Conclusion

- Perfect security is impossible
- Constant struggle to ensure *everything* is correct; attacker just has to find a *single* flaw
- We do our best using **systematic thinking** guided by models, e.g.:
  - The CIA triad
  - The information security model (asset/vulnerability/threat/attack)
  - Security strategy model (specify/implement/evaluate)
  - Attack surface modeling
  - Threat modeling (asset/vulnerability/attacker)
- Reduce likelihood of missing something with design principles, e.g.:
  - FIPS 200 security requirements (human and technical factors alike!)
  - Design principles for security in software design