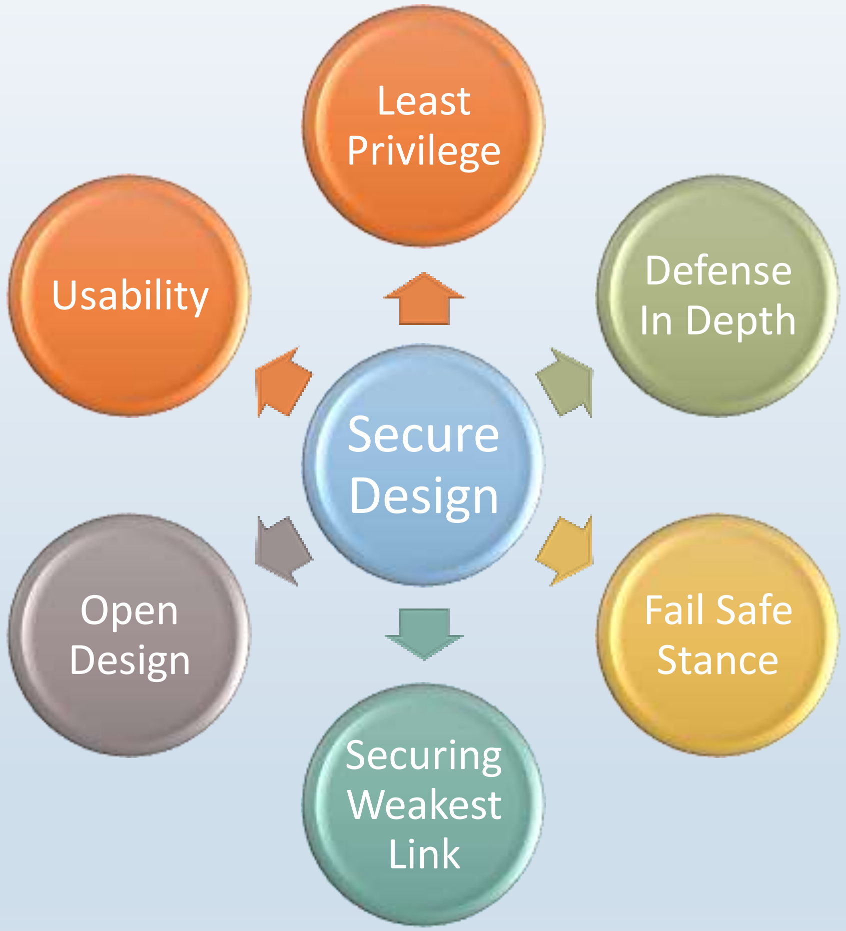
Secure Design Principles

The Foundation of System Security

Secure design principles, originally defined by Saltzer and Schroeder in 1975, are timeless rules for designing secure computing systems.

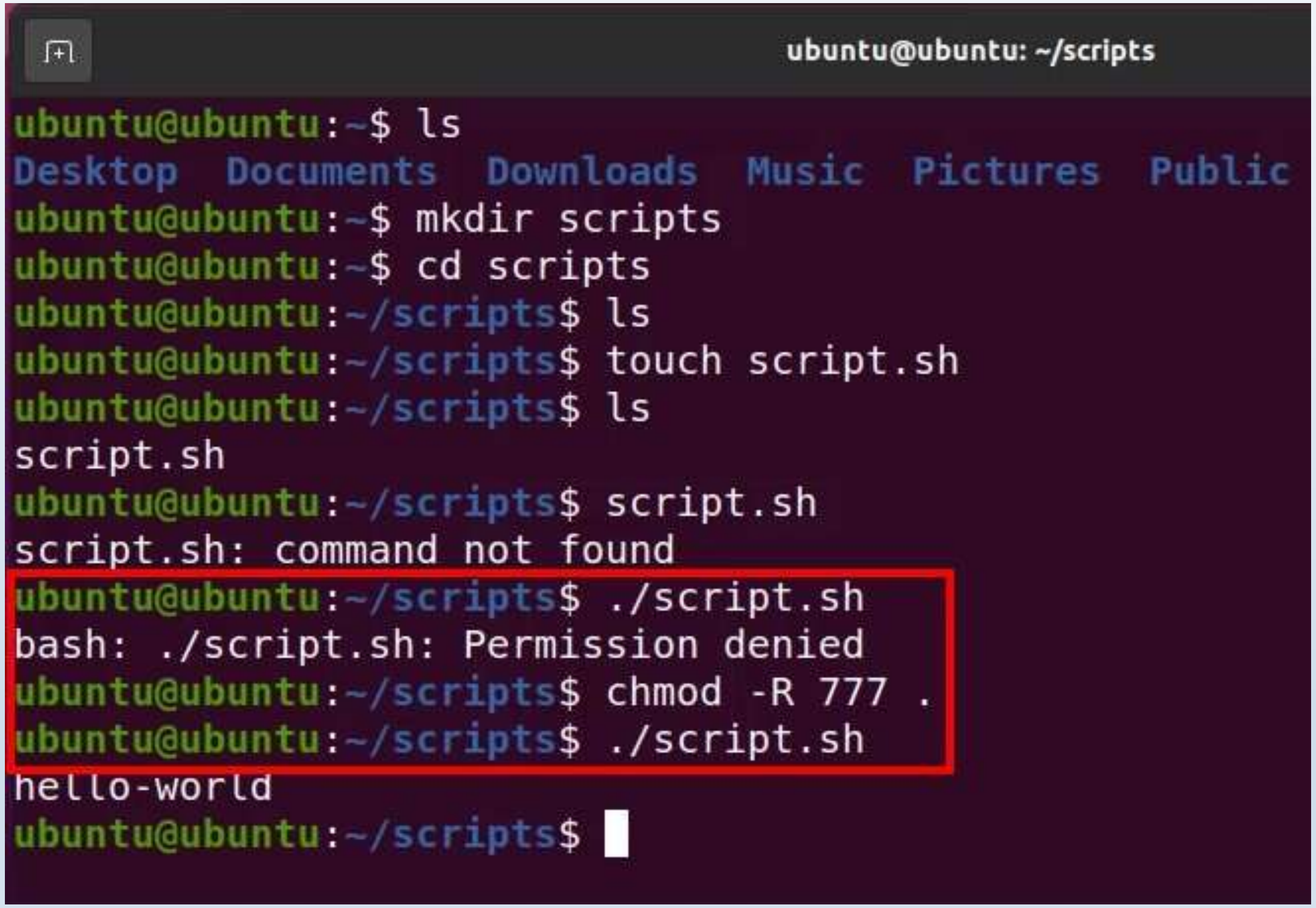


The Principle of Least Privilege

- Each user, process, or system component should operate with **only the permissions it absolutely needs** — nothing more.
- Limits users' access rights to only what are strictly required to do their jobs.

The Principle of Least Privilege - Examples

- In Linux, when you create a file, you are given read and write permission, but not execute
- Users who access a public website can read files from the server but can’t edit or upload files



ubuntu@ubuntu: ~/scripts

```
ubuntu@ubuntu:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public
ubuntu@ubuntu:~$ mkdir scripts
ubuntu@ubuntu:~$ cd scripts
ubuntu@ubuntu:~/scripts$ ls
ubuntu@ubuntu:~/scripts$ touch script.sh
ubuntu@ubuntu:~/scripts$ ls
script.sh
ubuntu@ubuntu:~/scripts$ script.sh
script.sh: command not found
ubuntu@ubuntu:~/scripts$ ./script.sh
bash: ./script.sh: Permission denied
ubuntu@ubuntu:~/scripts$ chmod -R 777 .
ubuntu@ubuntu:~/scripts$ ./script.sh
hello-world
ubuntu@ubuntu:~/scripts$ 
```

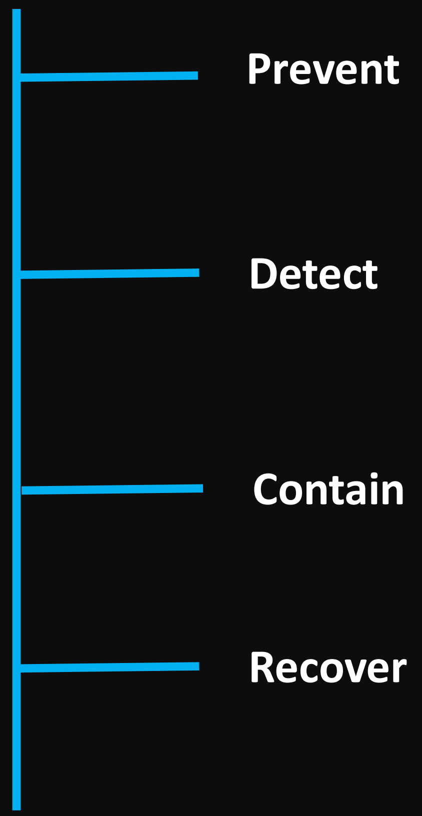
Benefits	Description
Reduces attack surface	Fewer entry points for attackers
Limits damage	If something breaks, it breaks in a small box
Improves accountability	Easier to track misuse

Why the PRINCIPLE of LEAST PRIVILEGE Matters
PoLP
SHRINK ATTACK SURFACE
IMPROVE operational PERFORMANCE
STOP MALWARE in its tracks
Prepare for AUDITS
guard against HUMAN ERROR

Defence-in-Depth - R*edundancy*

A strategy that leverages multiple security measures to protect an organization's assets.

If one line of defense is compromised, additional layers exist as a backup to ensure that threats are stopped along the way.



Defense in depth addresses the security vulnerabilities inherent not only with hardware and software but also with people, as <u>negligence or human error</u> are often the cause of a security breach.

Defense in Depth

Reference

https://www.fortinet.com/resources/cyberglossary/defense-in-depth#:~:text=Defense%20in%20depth%20is%20a,are%20stopped%20along%20the%20way.

- Physical controls
- Network security controls
- Administrative controls
- Antivirus
- Behavioral analysis

The Different Elements of a Defense-in-Depth System

- **Physical controls**: Key cards to enter a building or scanners to read fingerprints.
- **Network security controls**: This is software that authenticates an employee to enter the network and use a device or application.
- **Administrative controls**: This authorizes employees, once authenticated, to access only certain applications or parts of the network.
- **Antivirus**: This is the tool that stops malicious software from entering the network and spreading.
- **Behavioral analysis**: Algorithms and ML can detect anomalies in the behavior of employees and in the applications and devices themselves.

Diversity of Defense

- The idea behind *diversity of defense* is that using security systems from different vendors may reduce the chances of a common bug or configuration error that compromises them all.
- There is a tradeoff in terms of complexity and cost, however.
- Procuring and installing multiple different systems is going to be more difficult, take longer, and be more expensive than procuring and installing a single system (or even several identical systems). You're going to have to buy the multiple systems (at reduced discounts from each vendor, because you're buying less from them) and multiple support contracts to cover them. It's also going to take additional time and effort for your staff to learn how to deal with these different systems.

Securing the Weakest Link

- The weakest link is the part of a system that is the most vulnerable, susceptible, or easiest to attack.
- Weak Passwords
- People
- Implementation Vulnerabilities

- A **forgotten default password**
- A **user clicking a phishing email**
- A **server with outdated software**
- A **shared admin password**
- An **insecure mobile app** connecting to a secure backend

Securing the Weakest Link

Scenario	Weakest Link
Secure server but default admin credentials not changed	Credential mismanagement
Advanced encryption but poor key storage	Key management
Enforced password policy but no user training	Human factor
Web app is secure, but an API leaks data	Third-party exposure

How to Secure the Weakest Link

- **Identify all components**: devices, users, networks, software, cloud, physical access.Weak Passwords
- **Evaluate each for risk**: What could go wrong? Who could attack it?
- **Harden every link**:
  - Enforce strong passwords
  - Patch outdated systems
  - Train users (security awareness)
  - Use role-based access control
  - Limit trust in third-party components
- **Monitor and update**: What was secure last year may now be outdated or exposed.

Fail-Safe Stance

- Another fundamental principle of security is that, to the extent possible, systems should fail safe ; that is, if they're going to fail, they should fail in such a way that they deny access to an attacker, rather than letting the attacker in.
- The failure may also result in denying access to legitimate users as well, until repairs are made, but this is usually an acceptable trade-off.
- **Fail-safe stance involves designing a system in such a way that even if one or more components fail, you can still ensure some level of security**

Fail-Safe Stance

- The biggest application of this principle in network security is in choosing your site's stance with respect to security.
- Your stance is, essentially, your site's overall attitude towards security.
- Do you lean towards being restrictive or permissive? Are you more inclined to err in the direction of safety (some might call it paranoia) or freedom?

Fail-Safe Stance

- ***The default deny stance*** Specify only what you allow and prohibit everything else.
- ***The default permit stance*** Specify only what you prohibit and allow everything else.
- Example: if a packet filtering router goes down, it doesn't let any packets in

Secure by Default

- By default, be optimized for security
- A product's default configuration should prioritize maximum security.
- Users are protected immediately upon using the product.
- This feature is particularly important for users lacking technical expertise or time to configure complex security settings.
- ***Example : If you take a firewall out of the box and pop it onto a network, it only has one rule, deny everything.***

Simplicity - Simple systems are easier to test, audit, and secure.

- Keeping software as simple as possible is another way to preserve software security.
- Complex software is likely to have many more bugs and security holes than simple software.
- Code should be written so that it is possible to test each function in isolation.

Usability

- **Security features should be understandable, accessible, and user-friendly** — so people can actually use them correctly and consistently without needing deep technical knowledge.

Usability

- Many systems fail **not because they lack security features**, but because:
    - Users don’t understand them
    - They’re too complex or inconvenient
    - People disable or avoid them

Poor usability leads to **security failures** — even when the system is technically secure.

Security Features Do Not Imply Security

- Using one or more security features in a product does not ensure security
- A system’s security is not solely dependent upon the utilization of security features in its design, such as the encryption of passwords, but also depends on how it is used.