

Alle Angaben ohne Gewähr. Keine Garantie auf Vollständigkeit oder Richtigkeit.

1	Introduction	2
1.1	What is Access Control?	2
1.1.1	Definition according to OSI Basic Reference Model (1989)	2
1.1.2	Aspects of Access Control	2
1.2	Creating and enforcing policies	2
2	Foundations	2
2.1	Design Principles	2
2.1.1	Design Principles of Saltzer and Schroeder (1975)	2
2.1.2	A Contemporary Look at Saltzer and Schroeder's 1975 Design Principles	2
2.2	Policy Administration: Who is in charge of setting policies?	3
2.2.1	Discretionary Access Control (DAC)	3
2.2.2	Mandatory Access Control (MAC)	3
2.3	Access Control Matrix (ACM)	3
2.3.1	Primitive Operations	3
2.3.2	Commands	3
2.3.3	Commands: Example	4
2.3.4	Leak, Safety, Safety Question	4

1 Introduction

1.1 What is Access Control?

1.1.1 Definition according to OSI Basic Reference Model (1989)

- **Access Control:** “The prevention of unauthorized use of a resource”
- **Authorization:** “The granting of rights, which includes the granting of access based on access rights”
- Listed as a **security service:** “This service provides protection against unauthorized use of resources”
- Based on **policies:** “The control of access will be in accordance with various security policies”

1.1.2 Aspects of Access Control

- **Policy:** Focus on models, specification of security policies (What does it mean to be secure?)
- **Enforcement:** Focus on technologies, design and implementation of systems (How do we make it secure?)

1.2 Creating and enforcing policies

Classification by [Nobi, 2022]:

- **Policy Authoring/Engineering:** Define a policy model and access control rules
- **Policy Verification and Testing:** Test for leaks or contradictory privileges
- **Policy Administration:** Maintain and update policies/configurations
- **Policy Enforcement**

2 Foundations

2.1 Design Principles

2.1.1 Design Principles of Saltzer and Schroeder (1975)

Saltzer and Schroeder [Saltzer and Schroeder, 1975] proposed the following design principles for secure systems:

- **Economy of mechanism:** Keep the design as simple and small as possible
- **Fail-safe defaults:** Base access decisions on permission, not exclusion (default is lack of access)
- **Complete mediation:** Every access to every object must be checked
- **Open design:** Security through obscurity is not a good idea
- **Separation of privilege:** Divide access rights into multiple parts
- **Least privilege:** Give programs and users only the permissions they need for the job
- **Least common mechanism:** Minimize shared mechanisms
- **Psychological acceptability:** Make security mechanisms easy to use and understand

2.1.2 A Contemporary Look at Saltzer and Schroeder's 1975 Design Principles

Richard Smith [Smith, 2012] took a contemporary look at the design principles and found that most of them are still relevant today. He listed the following principles:

- **Continuous improvement:** Security is a process, not a product
- **Least privilege**
- **Defense in depth:** Use multiple layers of security

- **Open design**
- **Chain of control**: Ensure that trustworthy software is being executed
- **Deny by default**: Default should be lack of access
- **Transitive trust**: Trust should be transitive
- **Separation of duty**: Critical tasks should be divided among multiple individuals or entities

2.2 Policy Administration: Who is in charge of setting policies?

2.2.1 Discretionary Access Control (DAC)

In **Discretionary Access Control (DAC)** each resource is assigned an owner. The owner can decide who has access to the resource.

2.2.2 Mandatory Access Control (MAC)

In **Mandatory Access Control (MAC)** access is controlled by the system, not the owner. The system enforces access control based on a system-wide policy.

2.3 Access Control Matrix (ACM)

The **Protection State** of a system consists of all components and their privileges over each other. The **Access Control Matrix (ACM)** encodes the protection state.

	s_1	...	s_n	o_1	...	o_m
s_1						
\vdots						
s_n						

- Subjects $S = \{s_1, \dots, s_n\}$
- Objects $O = S \cup \{o_1, \dots, o_m\}$
- Rights $R = \{r_1, \dots, r_k\}$
- Entries $A[s_i, o_j] \subseteq R$

2.3.1 Primitive Operations

- **create subject s ; create object o** (Creates new column/row)
- **destroy subject s ; destroy object o** (Removes column/row)
- **enter r into $A[s, o]$** (Adds right r for subject s over object o)
- **delete r from $A[s, o]$** (Removes right r for subject s over object o)

Modification of the ACM usually requires executing a sequence of operations.

2.3.2 Commands

command <name>(<parameters>)

if

 <conditions>

then

 <operations>

end

If all **conditions** are met, all **operations** (conjunction \wedge) are executed, otherwise nothing happens.

2.3.3 Commands: Example

```
command make.file(p, f)
  create object f;
  center own into A[p, f];
  enter r into A[p, f];
  enter w into A[p, f];
end
```

2.3.4 Leak, Safety, Safety Question

- **Leak:** Adds a right r where there was not one before is called *leaking*
- **Safety:** If a system S , beginning in initial state s_0 cannot leak right r , it is safe with respect to the right r
- **Safety Question:** Does there exist **one** algorithm whether an **arbitrary** protection system S with initial state s_0 is safe with respect to a generic right r ?

The Safety Question is decidable for **mono-operational** commands. In general, the Safety Question is undecidable.

References

- [Nobi, 2022] Nobi, M. N. (2022). *Towards Machine Learning Based Access Control*. PhD thesis, The University of Texas at San Antonio.
- [Saltzer and Schroeder, 1975] Saltzer, J. and Schroeder, M. (1975). The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308.
- [Smith, 2012] Smith, R. E. (2012). A contemporary look at saltzer and schroeder’s 1975 design principles. *IEEE Security and Privacy*, 10(6):20–25.