

Chapter 3 FOUNDATIONAL RESULTS

- In 1976, Harrison, Ruzzo, and Ullman proved that in the most general abstract case, the security of computer systems was undecidable.
- Under what conditions can a generic algorithm determine whether a systems is secure?

3.1 The General Question

- Given a computer system, how can we determine if it is secure?
- Is there a generic algorithm that allows us to determine whether a computer system is secure?
 - If so, we could apply that algorithm to any system.
 - Although the algorithm might not tell us where the security problems were, it would tell us whether any existed.
- How is it possible to define a *secure* system?
 - For a general result, the definition should be as broad as possible.
- Let R be the set of generic (primitive) rights of the system.
- Definition 3-1 When a generic right r is added to an element of the ACM not already containing r , that right is said to be leaked.
- Our policy defines the authorized set of states A to be the set of states in which no command $c(x_1, \dots, x_n)$ can leak r . This implies that no generic rights can be added to the ACM.

Chapter 3 FOUNDATIONAL RESULTS

- Let a computer system begin in protection state s_0 .
- **Definition 3-2** If a system can never leak the right r , the system (including the initial state s_0) is called *safe* with respect to the right r . If the system can leak the right r (i.e., enter an unauthorized state), it is called *unsafe* with respect to the right r .
- The terms safe and unsafe are used because safety refers to the abstract model and security refers to the actual implementation.
- **EXAMPLE:** A computer system
 - Allows the network administrator to read all network traffic.
 - Disallows all other users from reading this traffic.
 - The system is designed in such a way that the network administrator cannot communicate with other users.
 - So, it is not possible for the right r of the network administrator to leak.
 - Hence, the system is secure!
 - Nevertheless, the OS has a flaw.
 - If a user specifies a certain file name in a file deletion system call, that user can obtain access to any file on the system, bypassing all file system access controls.
 - This is an implementation flaw, not a theoretical one!
 - Hence, the system is not secure!
- The question is (the *safety* question): Does there exist an algorithm for determining whether a given protection system with initial state s_0 is safe with respect to a generic right r ?

Chapter 3 FOUNDATIONAL RESULTS

3.2 Basic Results

- The simplest case is a system in which the commands are mono-operational.
- **Theorem 3-1** There exists an algorithm that will determine whether a given mono-operational protection system with initial state s_0 is safe with respect to a generic right r .
- Therefore, the system in the above theorem is *safe!*
- **Theorem 3-2** It is undecidable whether a given state of a given protection system is safe for a given generic right.
- Therefore, the system in the above theorem is *unsafe!*
- **Theorem 3-3** The set of unsafe systems is recursively enumerable.
- Assume that the *create* primitive is disallowed.
 - The safety question is decidable.
 - As no new subjects or objects are created, no new rights can be added to any element of the ACM.
 - If the leak has not occurred yet, it cannot occur.
- **Theorem 3-4** For protection systems without the create primitives, the question of safety is complete in P-SPACE.
- If deleting the create primitive makes the safety question decidable, would deleting *delete* and *destroy* primitives but not the create primitive also make the safety question decidable?
 - Such systems are called *monotonic* because they only increase (and not decrease) in size and complexity

Chapter 3 FOUNDATIONAL RESULTS

- **Theorem 3-5** It is undecidable whether a given configuration of a given monotonic protection system is safe for a given generic right.
- Restricting the number of conditions in the commands to two does not help.
- **Theorem 3-6** The safety question for biconditional monotonic protection systems is undecidable.
- If at most one condition per command is allowed, we have the following theorem.
- **Theorem 3-7** The safety question for monoconditional monotonic protection systems is decidable.
- This result can be made stronger.
- **Theorem 3-8** The safety question for monoconditional protection systems with *create*, *enter*, and *delete* primitives (but no *destroy* primitive) is decidable.
- The safety question is undecidable for generic protection models.
- The safety question is decidable if the protection system is restricted in some way.