Table of Contents

Security Models

Developed primarily between the 1970s and 1990s, these models provide formal, mathematically defined rules for controlling how information is accessed, modified, and allowed to flow within a system. Unlike informal policies, classic security models define what it means for a system to be secure and allow designers to prove that security properties—such as confidentiality and integrity—are preserved across all permitted system states and transitions.

At their core, these models address two fundamental security objectives. Confidentiality models, such as Bell–LaPadula, Information Flow, Noninterference, Lattice-based models, and Brewer–Nash, are designed to prevent unauthorized disclosure of information. They define strict rules governing how data may be read or shared across classification boundaries, ensuring sensitive information does not flow to unauthorized subjects. In contrast, integrity models, including Biba and Clark–Wilson, focus on preventing unauthorized or improper modification of data, preserving the accuracy and trustworthiness of information over time

Several models emphasize system behavior and state correctness rather than specific access decisions. State Machine models define security in terms of allowed transitions between secure system states, asserting that a system remains secure if every transition preserves defined security properties. Information Flow and Noninterference models extend this concept by examining how data moves through a system and ensuring that actions at one security level cannot influence or leak information to another in unintended ways.

Other models focus on authorization mechanics and rights management. The Take-Grant, Graham–Denning, and Harrison–Ruzzo–Ullman (HRU) models formalize how access rights can be created, transferred, delegated, and revoked. These models provide the theoretical basis for understanding discretionary access control, privilege propagation, and administrative authority within complex systems.

Together, these classic data security models establish a comprehensive framework for reasoning about who may access information, how information may flow, and under what conditions system state remains secure. While originally conceived for military, government, and high-assurance systems, their principles continue to underpin modern security controls, regulatory frameworks, and architectural decisions across enterprise and cloud environments. Understanding these models is essential for security architects and practitioners seeking to design systems with provable guarantees of confidentiality and integrity, rather than relying solely on ad hoc controls or assumptions.

State Machine

The State Machine Security Model is not a named model authored by one person. Instead, it is a general theoretical approach (automata theory) that comes from finite state machine theory in computer science from the 1950s, which was later applied to computer security by multiple researchers. The State Machine security model does not have a single creation date, but it emerged in computer Security in the early 1970. While no one "invented" the model, Morrie Gasser is widely cited for formalizing and explaining the State Machine approach in the context of computer security in his 1988 book "Building a Secure Computer System".

The State Machine model is not usually implemented directly. Instead, it is used to:

- Prove that a security system is correct
- Forms the foundation of other security models
- Formally verify security policies

The main concepts of the State Machine Security Model consist of:

1. A system is viewed as a set of states
2. Security is defined in terms of secure states
3. Systems change through state transitions
4. All state transitions must preserve security
5. The model focuses on security invariants
6. The State Machine model is foundational, not prescriptive
7. Emphasis is on formal verification

The State Machine security model ensures a system is secure by proving that every allowed action keeps the system in a secure state.

Graham–Denning

The Graham–Denning security model was introduced in 1972 by Peter J. Graham and Roger R. Denning in the article "Protection: Principles and Practice". This model is a formal access-control model that defines safe rules for creating, deleting, and managing users and resources, as well as granting, transferring, and revoking permissions. It defines who is allowed to give, take, or remove access to data and systems—and under what rules—so access does not get out of control. The Graham–Denning model focuses on how permissions themselves are managed. It was one of the earliest models for managing permissions over time.

The Graham–Denning model was created to define safe administrative rules so that:

- permissions cannot be created arbitrarily
- only authorized entities can change access rights
- the system remains secure as users and resources change.

The model defines eight basic operations that control all access changes:

1. Create an object (make a new file or resource)
2. Delete an object (remove a file or resource)
3. Create a subject (add a new user or process)
4. Delete a subject (remove a user or process)
5. Grant access rights (give someone permission)
6. Delete access rights (remove permission)
7. Transfer access rights (move permission to another subject)
8. Read access rights (check what permissions exist)

These rules ensure only authorized subjects can change permissions.

Bell–LaPadula

This Bell-LaPadula Model was published in a 1973 article by Scientists David Elliot Bell and Leonard J. LaPadula "Secure Computer Systems: Mathematical Foundations". They developed the model as part of a research effort to formalize U.S. Department of Defense (DoD) multilevel security (MLS) policy, with the goal of preventing unauthorized disclosure of classified information in shared computer systems.

Before Bell–LaPadula, computers used by governments and the military often had to be separated by security level (one computer for Secret data, another for Top Secret). As shared systems became common, there was a serious risk that highly classified information could accidentally flow to lower-cleared users.
Bell–LaPadula was designed to formally prove that a system could safely handle multiple security levels at once without leaking secrets, even as users and files changed over time.

To enforce this, the model assigns:

- Clearance levels to people (users or programs)
- Classification levels to data (files, records, documents)

1. No Read Up (Simple Security Property)
A user cannot read information that is classified higher than their clearance.
A person with Confidential clearance cannot read a Top Secret document.
This rule prevents unauthorized access to sensitive information.

2. No Write Down (*-Property / Star Property)
A user cannot write information to a lower classification level.
A person with Top Secret clearance cannot copy or save Top Secret information into a Confidential file.
This rule prevents accidental or intentional leaks of sensitive information to less secure locations.

Harrison–Ruzzo–Ullman (HRU)

The Harrison–Ruzzo–Ullman (HRU) security model was created by Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman. It was released in 1976 in the article

"Protection in Operating Systems". The Harrison–Ruzzo–Ullman (HRU) security model is a formal access-control model that explains how permissions (access rights) in a computer system can change over time and whether those changes can lead to unauthorized access. The HRU model is especially known for formally defining the term "safe" in access-control systems. A system is considered "safe" with respect to a specific right if no sequence of allowed operations can ever grant that right to an unauthorized subject. This model is used to formally analyze how permissions evolve and to determine whether rights could "leak" in ways system designers did not intend.

The HRU model represents a system using a table (matrix):

- Rows - users or programs (subjects)
- Columns - files, devices, or even other users (objects)
- Cells - permissions (like read, write, delete)

This matrix formally captures who can do what to which resource at any moment in time.

The HRU model defines six basic operations that can change permissions:

- Create a subject (add a user or process)
- Delete a subject
- Create an object (add a file or resource)
- Delete an object
- Grant a permission (add a right)
- Revoke a permission (remove a right)

These operations model how permissions evolve in real systems through administrative and user actions.

Lattice Based

The first formal definition of the Lattice-Based security model was created by Dorothy E. Denning in 1976 in the article "A Lattice Model of Secure Information Flow". The Lattice-Based security model is a formal way to control who can access or move data by arranging users and data into ordered levels and only allowing information to flow in approved directions. It ensures that information can move only in allowed directions between security levels, and never in forbidden directions. This model is used to prevent sensitive data from leaking to less-trusted users or systems, even indirectly. This work unified earlier security ideas under a single mathematical framework. Denning created the lattice model to formally describe all allowed and disallowed paths that information could take, including indirect flows through programs, users, or systems.

The model works by giving everything a label:

- Users get clearance labels
- Data gets classification labels

These labels are arranged in a structured hierarchy called a "lattice".

Example labels:

- Public
- Confidential
- Secret
- Top Secret

The lattice model guarantees:

- No unauthorized information flow
- Consistent enforcement of security rules
- Formal, provable security properties.

Take-Grant

The Take-Grant model was introduced in a 1977 article by Richard J. Lipton and Lawrence Snyder "A Linear Time Algorithm for Deciding Subject Security". The Take-Grant security model is a formal access-control and protection model that explains how permissions (rights) can spread through a computer system over time. Rather than focusing on secrecy levels (like *Top Secret* vs *Public*), the Take-Grant model focuses on how access rights move between users and objects.

The Take-Grant model was created specifically to analyze the safety of a system—that is, whether rights can leak in ways the system designer did not intend.

The Take-Grant model represents a system as a diagram (graph):

Dots (nodes) = users or things (files, programs)
Arrows (edges) = permissions
Labels on arrows = what permission exists
Take = Allows a user to take permissions from another entity.
Grant = Allows a user to give their permissions to someone else.

These two permissions control how all other permissions spread.

Biba

This Model was invented by Scientist Kenneth J. Biba. He introduced the model in 1977 in his paper “Integrity Considerations for Secure Computer Systems”, published while he was at The MITRE Corporation.

The Biba Integrity Model is a security model (a set of formal rules) designed to protect data integrity, meaning it helps ensure information stays accurate and trustworthy by controlling who (or what software) can read or change it. The model is used as a conceptual foundation for systems where unauthorized modification (tampering) is the major risk, such as environments requiring strong integrity controls. The Biba model is a way to protect data accuracy. Everything—users and files—gets a trust level. The rules are ‘no read down’ and ‘no write up.’ That means highly trusted processes can’t rely on lower-trust data, and lower-trust users can’t change higher-trust records. It’s used when preventing tampering and keeping records correct matters most.

No Read Down (Simple Integrity Property)
A subject (user/program) cannot read data at a lower integrity level than its own. If you’re a highly trusted worker/process, you shouldn’t base your decisions on lower-trust information.

No Write Up (Star * Integrity Property)
A subject cannot write/modify data at a higher integrity level than its own. If you’re less trusted, you can’t change more trusted records.

Information Flow

This concept was first brought up in 1976 article by Dorothy E. Denning “A Lattice Model of Secure Information Flow”. With that said the Information Flow model is most widely credited to Joseph A. Goguen and José Meseguer, who formally defined in a 1982 article “Security Policies and Security Models”. This model focuses on how data moves inside a system, not just who can access it. Sensitive information must not flow to places where it is not allowed. Unlike access-control models that only decide whether someone can open a file, the Information Flow model also controls what happens to the data after it has been accessed. This model tracks how information propagates through programs, processes, and systems. This model controls what happens to the data after it has been accessed. The system assures secret data doesn’t flow into public outputs. It also assures untrusted data doesn’t flow into trusted systems.

Information Flow follows the following steps:

1. Label the data - Each variable, file, or message is assigned a security label
2. Define allowed flows - Rules define which labels are allowed to send data to which other labels.

3. Block illegal flows - If a program tries to move data in a forbidden direction the action is blocked

Noninterference

This model was first defined by Joseph A. Goguen and José Meseguer in 1982 in the article "Security Policies and Security Models". It is considered one of the most important theoretical foundations of information-flow security. The Noninterference security model is a formal information-flow security model that ensures activities at a higher security level do not affect what can be observed at a lower security level. Someone who is not allowed to see secret data should not be able to learn anything about it—even indirectly—by watching how the system behaves.

This system has High-level and Low-level users. High-level is allowed to work with secret data, while low-level users are not allowed to see secret data. No matter what high-level users do, the low-level users must see the system behave exactly the same. If low-level users ever see different results, timing, outputs, or behavior based on secret actions, then information has leaked, and the system is not secure.

First you divide the system into two security levels including High for secret/sensitive data and Low for public/unclassified data. Then you allow low-level users to see only low-level outputs. The system is secure only if, changing high-level inputs never changes low-level outputs. If a low-level user can tell anything about what high-level users are doing, noninterference is violated.

Clark-Wilson

The Clark–Wilson security model was created by David D. Clark and David R. Wilson. They introduced the model in 1987 in a paper titled "A Comparison of Commercial and Military Computer Security Policies."

The Clark–Wilson model is a security model focused on data integrity, meaning it is designed to keep information accurate, consistent, and protected from improper changes. Instead of trying to keep information secret, the model's goal is to ensure that data is only changed in approved, correct ways.

Clark–Wilson was created to protect against:

- Accidental data errors
- Insider fraud
- Unauthorized or improper data modification

Core Concepts:

1. Well-formed transactions - A well-formed transaction is a carefully designed process that changes data from one correct state to another correct state.
2. Separation of duties - No single person is allowed to control an entire critical process.
3. Indirect access to data - Users never interact with important data directly. They can only use approved programs (called transformation procedures) to make changes.
4. Auditing and verification - Every important action is logged and checked so problems can be detected and investigated.

Brewer–Nash

The model was introduced in 1989 by David F. C. Brewer and Michael J. Nash in their paper "The Chinese Wall Security Policy".

The Brewer–Nash model is a security model designed to prevent conflicts of interest. Its goal is to ensure that a person who has access to one company's sensitive information cannot later access sensitive information from that company's competitors.

Because of this purpose, the model is also known as the Chinese Wall security model—an analogy for an invisible wall that blocks information from crossing between competing parties once access has been established.

Core Concepts:

1. Information is grouped by conflict - Data is grouped by company. Competing companies are grouped into a conflict-of-interest class
2. Initial access is allowed - When a person has not yet accessed any data, they may choose any company to work with.
3. A "wall" is created after access - Once the person accesses one company's data, access to competing companies' data is blocked from that point forward
4. Dynamic access control - Access rights change over time as users interact with data.

Conclusion

The classic data security models examined in this document represent the foundational thinking that shaped modern computer security. Developed primarily between the 1970s and 1990s, these models introduced formal, mathematically grounded approaches to defining and enforcing security properties such as confidentiality, integrity, and controlled information flow. Rather than relying on informal policy statements or ad hoc controls, these models established precise rules for how systems must behave to remain secure across all possible states and transitions.

Each model addresses a distinct aspect of the security problem space. State Machine models provide the underlying assurance framework by defining security as the preservation of invariants across system states. Confidentiality-focused models such as Bell–LaPadula, Lattice-Based security, Information Flow, Noninterference, and Brewer–Nash define strict constraints on how information may be observed or propagated. Integrity-oriented models, including Biba and Clark–Wilson, emphasize the correctness and trustworthiness of data and the processes that modify it. Authorization-centric models such as Graham–Denning, Harrison–Ruzzo–Ullman, and Take-Grant formalize how access rights are created, delegated, transferred, and revoked, enabling rigorous reasoning about privilege management and safety.

Although these models were originally conceived for military, government, and high-assurance systems, their influence extends far beyond their initial use cases. Concepts such as least privilege, separation of duties, mandatory access control, trusted subjects, and controlled information flow are directly traceable to these foundational works and remain central to modern operating systems, database platforms, cloud architectures, and regulatory frameworks. Even where the models are not implemented directly, they continue to provide the theoretical basis for evaluating security guarantees and identifying classes of design flaws that cannot be mitigated through implementation alone.

Understanding these classic data security models is essential for security architects, engineers, and risk professionals who seek to design systems with provable security properties rather than relying on assumptions or reactive controls. By studying these models and their original disclosures, practitioners gain not only historical context, but also a deeper appreciation of the enduring principles that underpin secure system design. In an era of increasingly complex and interconnected systems, these foundational models remain as relevant as ever, offering clarity, rigor, and discipline in the pursuit of trustworthy computing.