Architecture of Enterprise Applications VII Modifiability and Security

Haopeng Chen

***RE**liable, **IN**telligent and **S**calable Systems Group (**REINS**)*
Shanghai Jiao Tong University
Shanghai, China
e-mail: chen-hp@sjtu.edu.cn

MODIFIABILITY

- Modifiability is about the cost of change. It brings up two concerns.
  - What can change (the artifact)? A change can occur to any aspect of a system
    - the functions that the system computes
    - the platform the system exists on
    - the environment within which the system operates
    - the qualities the system exhibits
    - capacity

MODIFIABILITY

- When is the change made and who makes it (the environment)?Changes can be made to
  - the implementation (by modifying the source code)
  - during compile (using compile-time switches)
  - during build (by choice of libraries)
  - during configuration setup (by a range of techniques, including parameter setting)
  - during execution (by parameter setting).
- A change can also be made by a developer, an end user, or a system administrator.

- Once a change has been specified, the new implementation must be designed, implemented, tested, and deployed. All of these actions take time and money, both of which can be measured.

MODIFIABILITY

- Source of stimulus.
  - This portion specifies who makes the changes—the developer, a system administrator, or an end user. Clearly, there must be machinery in place to allow the system administrator or end user to modify a system, but this is a common occurrence.
- Stimulus.
  - This portion specifies the changes to be made.
- Artifact.
  - This portion specifies what is to be changed. the functionality of a system, its platform, its user interface, its environment, or another system with which it interoperates
- Environment.
  - This portion specifies when the change can be made. Design time, compile time, build time, initiation time, or runtime

MODIFIABILITY

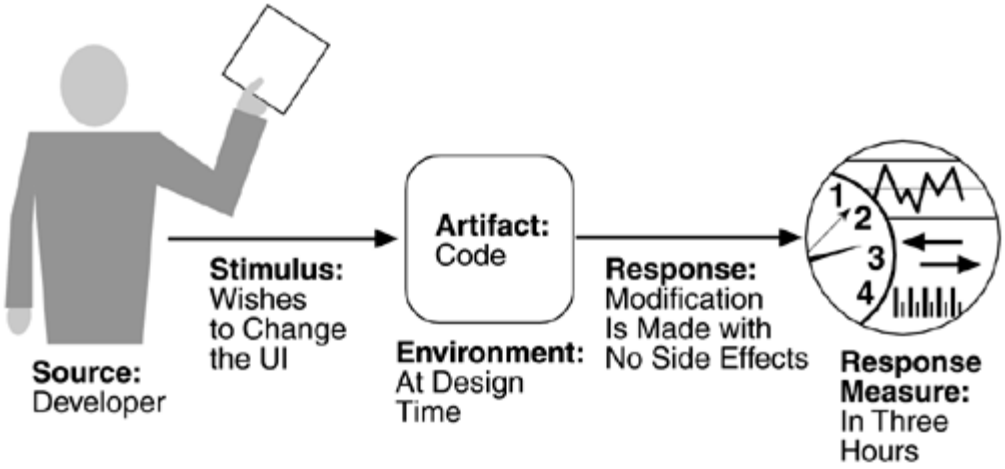
- Response.
  - Whoever makes the change must understand how to make it, and then make it, test it and deploy it.

- Response measure.
  - All of the possible responses take time and cost money, and so time and cost are the most desirable measures. Time is not always possible to predict, however, and so less ideal measures are frequently used, such as the extent of the change (number of modules affected).

MODIFIABILITY

Portion of Scenario	Possible Values
Source	End user, developer, system administrator
Stimulus	Wishes to add/delete/modify/vary functionality, quality attribute, capacity
Artifact	System user interface, platform, environment; system that interoperates with target system
Environment	At runtime, compile time, build time, design time
Response	Locates places in architecture to be modified; makes modification without affecting other functionality; tests modification; deploys modification
Response Measure	Cost in terms of number of elements affected, effort, money; extent to which this affects other functions or quality attributes

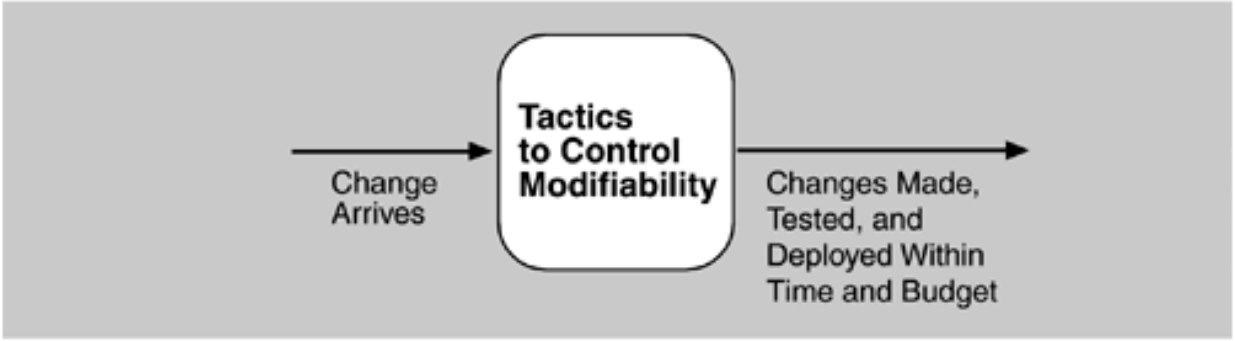
MODIFIABILITY



Modifiability Tactics

- Tactics to control modifiability have as their goal controlling the time and cost to implement, test, and deploy changes.

Goal of modifiability tactics



Modifiability Tactics

- We organize the tactics for modifiability in sets according to their goals.
  - One set has as its goal reducing the number of modules that are directly affected by a change. We call this set "localize modifications."
  - A second set has as its goal limiting modifications to the localized modules. We use this set of tactics to "prevent the ripple effect."
  - A third set of tactics has as its goal controlling deployment time and cost. We call this set "defer binding time."

Modifiability Tactics-localize modifications

- The goal of tactics in this set is
  - to assign responsibilities to modules during design such that anticipated changes will be limited in scope. We identify the following tactics.

- Maintain semantic coherence.
  - Semantic coherence refers to the relationships among responsibilities in a module.
    - The goal is to ensure that all of these responsibilities work together without excessive reliance on other modules.
  - Achievement of this goal comes from choosing responsibilities that have semantic coherence.
  - Instead, semantic coherence should be measured against a set of anticipated changes.
    - One sub-tactic is to abstract common services.

- Anticipate expected changes.
  - Considering the set of envisioned changes provides a way to evaluate a particular assignment of responsibilities.
  - The tactic of anticipating expected changes does not concern itself with the coherence of a module's responsibilities but rather with minimizing the effects of the changes.
  - In reality this tactic is difficult to use by itself since it is not possible to anticipate all changes. For that reason, it is usually used in conjunction with semantic coherence.

Modifiability Tactics-localize modifications

- Generalize the module.
  - Making a module more general allows it to compute a broader range of functions based on input.
  - The more general a module, the more likely that requested changes can be made by adjusting the input language rather than by modifying the module.

- Limit possible options.
  - Modifications, especially within a product line may be far ranging and hence affect many modules.
  - Restricting the possible options will reduce the effect of these modifications.

Modifiability Tactics-localize modifications

- 我们允许用户将RUBiS部署在不同的操作系统上，但是限定只能在Windows XP版本或以上，以及Ubuntu 8.04版本以上的操作系统中选择，这样就可以避免为了适应更多的操作系统而做出的额外修改。

- 我们在RUBiS中，就设计了Utilities包，用于放置所有的公共服务，这就是为了在将来修改这些公共服务时，将修改局限在这个包的范围内。

Modifiability Tactics-prevent the ripple effect

- A ripple effect from a modification is the necessity of making changes to modules not directly affected by it.

- We identify eight types of dependencies that one module can have on another :

1. Syntax of
   - data.
     - For B to compile (or execute) correctly, the type (or format) of the data that is produced by A and consumed by B must be consistent with the type (or format) of data assumed by B.
   - service.
     - For B to compile and execute correctly, the signature of services provided by A and invoked by B must be consistent with the assumptions of B.

Modifiability Tactics-prevent the ripple effect

2. Semantics of

- data.
  - For B to execute correctly, the semantics of the data produced by A and consumed by B must be consistent with the assumptions of B.
- service.
  - For B to execute correctly, the semantics of the services produced by A and used by B must be consistent with the assumptions of B.

3. Sequence of

- data.
  - For B to execute correctly, it must receive the data produced by A in a fixed sequence. For example, a data packet's header must precede its body in order of reception (as opposed to protocols that have the sequence number built into the data).
- control.
  - For B to execute correctly, A must have executed previously within certain timing constraints. For example, A must have executed no longer than 5ms before B executes.

Modifiability Tactics-prevent the ripple effect

4. Identity of an interface of A.

- A may have multiple interfaces.
  - For B to compile and execute correctly, the identity (name or handle) of the interface must be consistent with the assumptions of B.

5. Location of A (runtime).

- For B to execute correctly, the runtime location of A must be consistent with the assumptions of B.
  - For example, B may assume that A is located in a different process on the same processor.

6. Quality of service/data provided by A.

- For B to execute correctly, some property involving the quality of the data or service provided by A must be consistent with B's assumptions.
  - For example, data provided by a particular sensor must have a certain accuracy in order for the algorithms of B to work correctly.

Modifiability Tactics-prevent the ripple effect

7. Existence of A.

- For B to execute correctly, A must exist.
  - For example, if B is requesting a service from an object A, and A does not exist and cannot be dynamically created, then B will not execute correctly.

8. Resource behavior of A.

- For B to execute correctly, the resource behavior of A must be consistent with B's assumptions.
  - This can be either resource usage of A (A uses the same memory as B) or resource ownership (B reserves a resource that A believes it owns).

Modifiability Tactics-prevent the ripple effect

- Hide information.
  - Information hiding is the decomposition of the responsibilities for an entity (a system or some decomposition of a system) into smaller pieces and choosing which information to make private and which to make public.
  - The public responsibilities are available through specified interfaces.
  - The goal is to isolate changes within one module and prevent changes from propagating to others.
  - This is the oldest technique for preventing changes from propagating.

Modifiability Tactics-prevent the ripple effect

- Maintain existing interfaces.
  - If B depends on the name and signature of an interface of A, maintaining this interface and its syntax allows B to remain unchanged.
  - Of course, this tactic will not necessarily work if B has a semantic dependency on A, since changes to the meaning of data and services are difficult to mask.
  - Also, it is difficult to mask dependencies on quality of data or quality of service, resource usage, or resource ownership.
  - Interface stability can also be achieved by separating the interface from the implementation.
  - This allows the creation of abstract interfaces that mask variations.

Modifiability Tactics-prevent the ripple effect

- Patterns that implement this tactic include
  - adding interfaces. Most programming languages allow multiple interfaces.
  - adding adapter. Add an adapter to A that wraps A and provides the signature of the original A.
  - providing a stub A. If the modification calls for the deletion of A, then providing a stub for A will allow B to remain unchanged if B depends only on A's signature.

Modifiability Tactics-prevent the ripple effect

- Restrict communication paths.
  - Restrict the modules with which a given module shares data.
    - That is, reduce the number of modules that consume data produced by the given module and the number of modules that produce data consumed by it.
  - This will reduce the ripple effect since data production/consumption introduces dependencies that cause ripples.

- Use an intermediary.
  - If B has any type of dependency on A other than semantic, it is possible to insert an intermediary between B and A that manages activities associated with the dependency.
  - As before, in the worst case, an intermediary cannot compensate for semantic changes.

Modifiability Tactics-prevent the ripple effect

- The intermediaries are:
  - data (syntax).
    - Repositories (both blackboard and passive) act as intermediaries between the producer and consumer of data.
  - service (syntax).
    - The facade, bridge, mediator, strategy, proxy, and factory patterns all provide intermediaries that convert the syntax of a service from one form into another.
  - identity of an interface of A.
    - A broker pattern can be used to mask changes in the identity of an interface.
  - location of A (runtime).
    - A name server enables the location of A to be changed without affecting B.
  - resource behavior of A or resource controlled by A.
    - A resource manager is an intermediary that is responsible for resource allocation.
  - existence of A.
    - The factory pattern has the ability to create instances as needed, and thus the dependence of B on the existence of A is satisfied by actions of the factory.

Modifiability Tactics-prevent the ripple effect

- 在RUBiS提供的众多版本中，有一个版本包含一个消息驱动Bean构件，使得用户可以通过异步方式提交请求。
- 这个消息驱动Bean和客户端代码之间不直接进行交互，而是通过一个消息队列作为中介来交互。
- 这样当该消息驱动Bean在处理消息的具体逻辑发生变更时，客户端代码并不需要进行修改，因为它与消息队列交互的方式并未发生变化，这就有效地防止了涟漪效应。

Modifiability Tactics-defer binding time

- Our modifiability scenarios include two elements that are not satisfied by reducing the number of modules to be changed—time to deploy and allowing non-developers to make changes.
  - Deferring binding time supports both of those scenarios at the cost of requiring additional infrastructure to support the late binding.

- Many tactics are intended to have impact at lead-time or runtime, such as the following.
  - Runtime registration supports plug-and-play operation at the cost of additional overhead to manage the registration. Publish/subscribe registration, for example, can be implemented at either runtime or load time.
  - Configuration files are intended to set parameters at startup.
  - Polymorphism allows late binding of method calls.
  - Component replacement allows load time binding.
  - Adherence to defined protocols allows runtime binding of independent processes

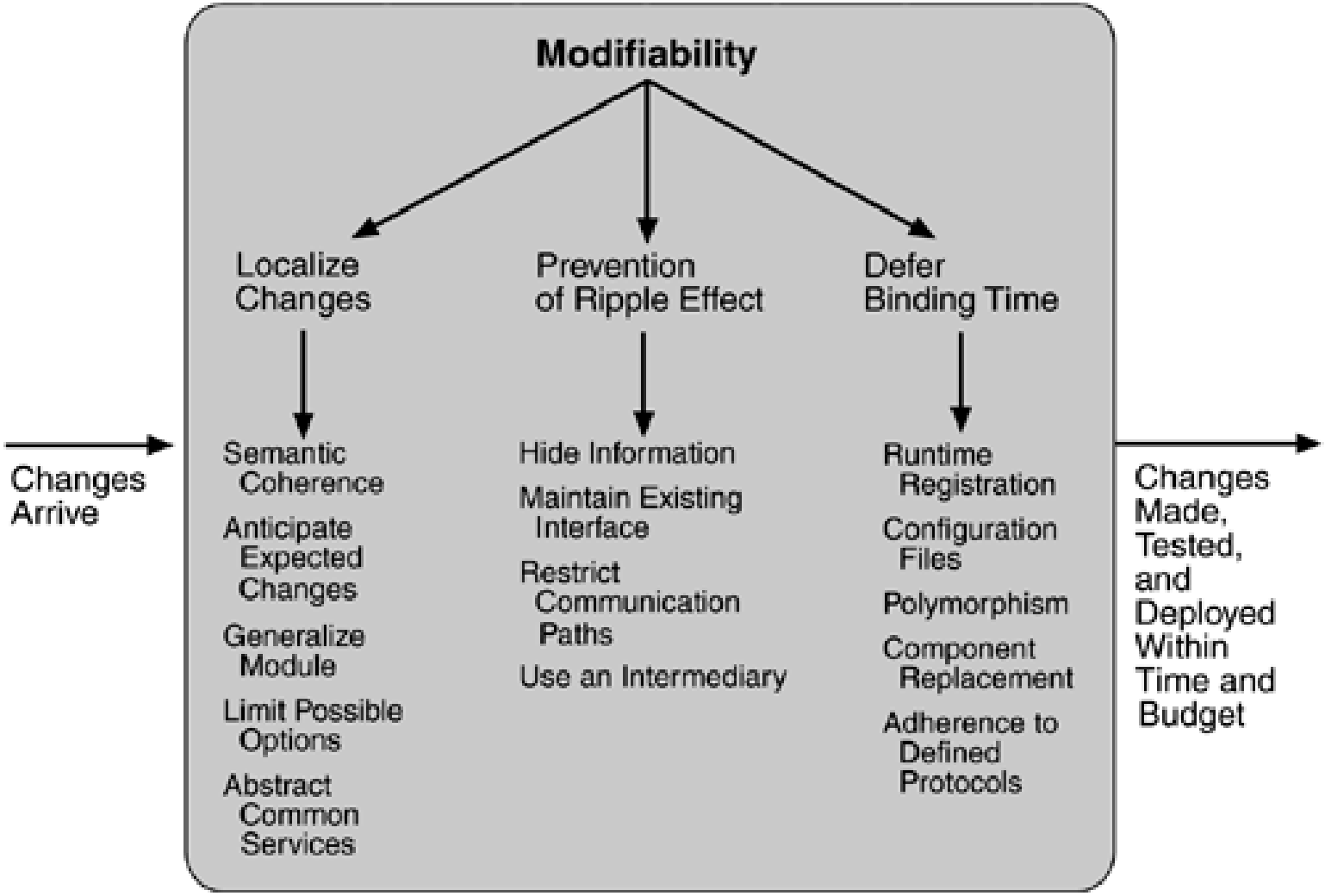
Modifiability Tactics-defer binding time

- 我们在RUBiS中提供了各种环境变量，它们在运行时绑定到命名与目录服务的名字树下，供程序调用。

- 上述RUBiS的环境变量就是在纯文本的部署说明符中声明的，我们可以通过文本编辑器修改它的值，使其在系统启动时对参数进行赋值，这使得非系统开发人员也可以做出这种变更。

- 在RUBiS中，我们设计了两个主键生成构件，一个生成UUID主键，一个生成自增主键。我们通过插件模式将其集成到系统中，并根据配置文件来确定具体加载哪一个构件。

Modifiability Tactics-Summary



SECURITY

- Security is a measure of the system's ability to resist unauthorized usage while still providing its services to legitimate users.
  - An attempt to breach security is called an attack and can take a number of forms.

- Security can be characterized as a system providing nonrepudiation, confidentiality, integrity, assurance, availability, and auditing.

SECURITY

- Nonrepudiation
  - is the property that a transaction (access to or modification of data or services) cannot be denied by any of the parties to it. This means you cannot deny that you ordered that item over the Internet if, in fact, you did.

- Confidentiality
  - is the property that data or services are protected from unauthorized access. This means that a hacker cannot access your income tax returns on a government computer.

- Integrity
  - is the property that data or services are being delivered as intended. This means that your grade has not been changed since your instructor assigned it.

SECURITY

- Assurance
  - is the property that the parties to a transaction are who they purport to be. This means that, when a customer sends a credit card number to an Internet merchant, the merchant is who the customer thinks they are.
- Availability
  - is the property that the system will be available for legitimate use. This means that a denial-of-service attack won't prevent your ordering this book.
- Auditing
  - is the property that the system tracks activities within it at levels sufficient to reconstruct them. This means that, if you transfer money out of one account to another account, in Switzerland, the system will maintain a record of that transfer.

SECURITY

- Source of stimulus.
  - The source of the attack may be either a human or another system. It may have been previously identified (either correctly or incorrectly) or may be currently unknown.
  - The attack itself is unauthorized access, modification, or denial of service.
- Stimulus.
  - The stimulus is an attack or an attempt to break security.
  - We characterize this as an unauthorized person or system trying to display information, change and/or delete information, access services of the system, or reduce availability of system services.
- Artifact.
  - The target of the attack can be either the services of the system or the data within it.
- Environment.
  - The attack can come when the system is either online or offline, either connected to or disconnected from a network, either behind a firewall or open to the network.

- Response.
  - Using services without authorization or preventing legitimate users from using services is a different goal from seeing sensitive data or modifying it.
  - Thus, the system must authorize legitimate users and grant them access to data and services, at the same time rejecting unauthorized users, denying them access, and reporting unauthorized access.

- Response measure.
  - Measures of a system's response include the difficulty of mounting various attacks and the difficulty of recovering from and surviving attacks.

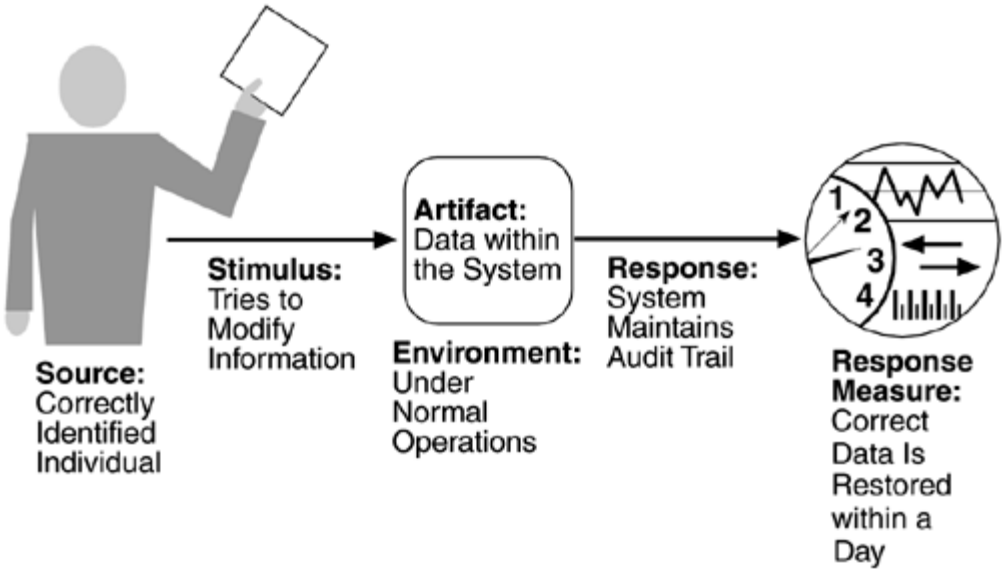
SECURITY

Portion of Scenario	Possible Values
Source	Individual or system that is correctly identified, identified incorrectly, of unknown identity who is internal/external, authorized/not authorized with access to limited resources, vast resources
Stimulus	Tries to display data, change/delete data, access system services, reduce availability to system services
Artifact	System services; data within system
Environment	Either online or offline, connected or disconnected, fire-walled or open

SECURITY

Response	**Authenticates user; hides identity of the user; blocks access to data and/or services; allows access to data and/or services; grants or withdraws permission to access data and/or services; records access/modifications or attempts to access/modify data/services by identity; stores data in an unreadable format; recognizes an unexplainable high demand for services, and informs a user or another system, and restricts availability of services**
Response Measure	**Time/effort/resources required to circumvent security measures with probability of success; probability of detecting attack; probability of identifying individual responsible for attack or access/modification of data and/or services; percentage of services still available under denial-of-services attack; restore data/services; extent to which data/services damaged and/or legitimate access denied**

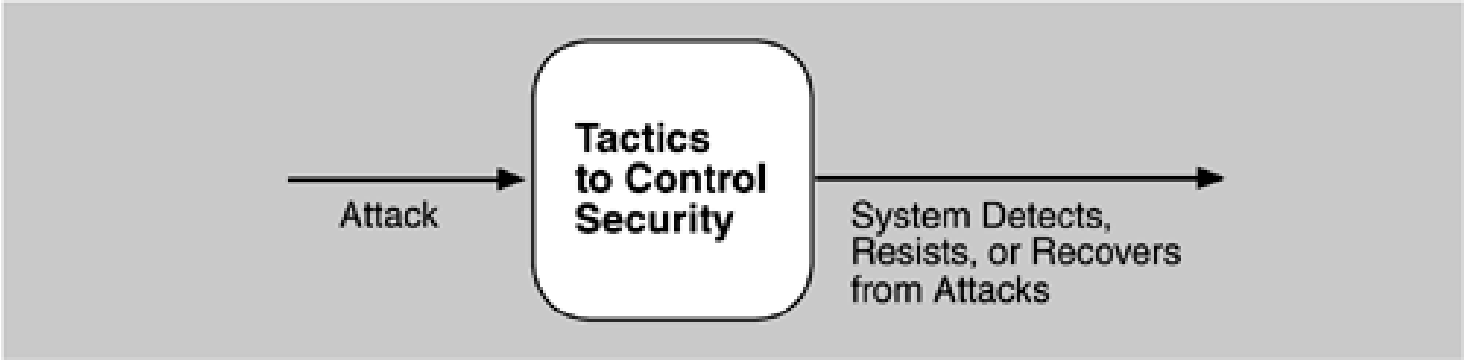
SECURITY


Source:
Correctly
Identified
Individual
Stimulus:
Tries to
Modify
Information
Artifact:
Data within
the System
Environment:
Under
Normal
Operations
Response:
System
Maintains
Audit Trail
1
2
3
4
Response
Measure:
Correct
Data Is
Restored
within a
Day

Security Tactics

- Tactics for achieving security can be divided into those concerned with
  - resisting attacks
  - detecting attacks
  - recovering from attacks.
- Using a familiar analogy
  - putting a lock on your door is a form of resisting an attack,
  - having a motion sensor inside of your house is a form of detecting an attack,
  - and having insurance is a form of recovering from an attack.

Goal of security tactics



Security Tactics-resisting attacks

- we identified nonrepudiation, confidentiality, integrity, and assurance as goals in our security characterization.

- The following tactics can be used in combination to achieve these goals.
  - Authenticate users.
  - Authorize users.
  - Maintain data confidentiality.
    - Encryption
    - Communication links
    - virtual private network (VPN)
    - Secure Sockets Layer (SSL)
  - Maintain integrity.
  - Limit exposure.
  - Limit access.
    - Firewalls

Security Tactics-resisting attacks

- 我们在部署RUBiS时，可以通过防火墙设置只有8080端口可访问，通过其他端口访问RUBiS的请求都会被防火墙拒绝。对于通过合法端口访问的用户，我们也进行了认证与授权，而其中用户密码是加密之后存储到数据库中的。当用户提交竞价信息时，为了防止数据被网络上的黑客截获被篡改，竞价信息将被签名之后发送到服务器端。

Security Tactics-detecting attacks

- The detection of an attack is usually through an intrusion detection system.
  - Such systems work by comparing network traffic patterns to a database.
  - In the case of misuse detection, the traffic pattern is compared to historic patterns of known attacks.
  - In the case of anomaly detection, the traffic pattern is compared to a historical baseline of itself.
  - Frequently, the packets must be filtered in order to make comparisons.
  - Filtering can be on the basis of protocol, TCP flags, payload sizes, source or destination address, or port number.

- Intrusion detectors must have some sort of
  - sensor to detect attacks,
  - managers to do sensor fusion,
  - databases for storing events for later analysis,
  - tools for offline reporting and analysis,
  - and a control console so that the analyst can modify intrusion detection actions.

Security Tactics-detecting attacks

- 我们可以在部署RUBiS系统的服务器上安装入侵检测系统，并将其探测器安装到系统中需要被检测的部件上。

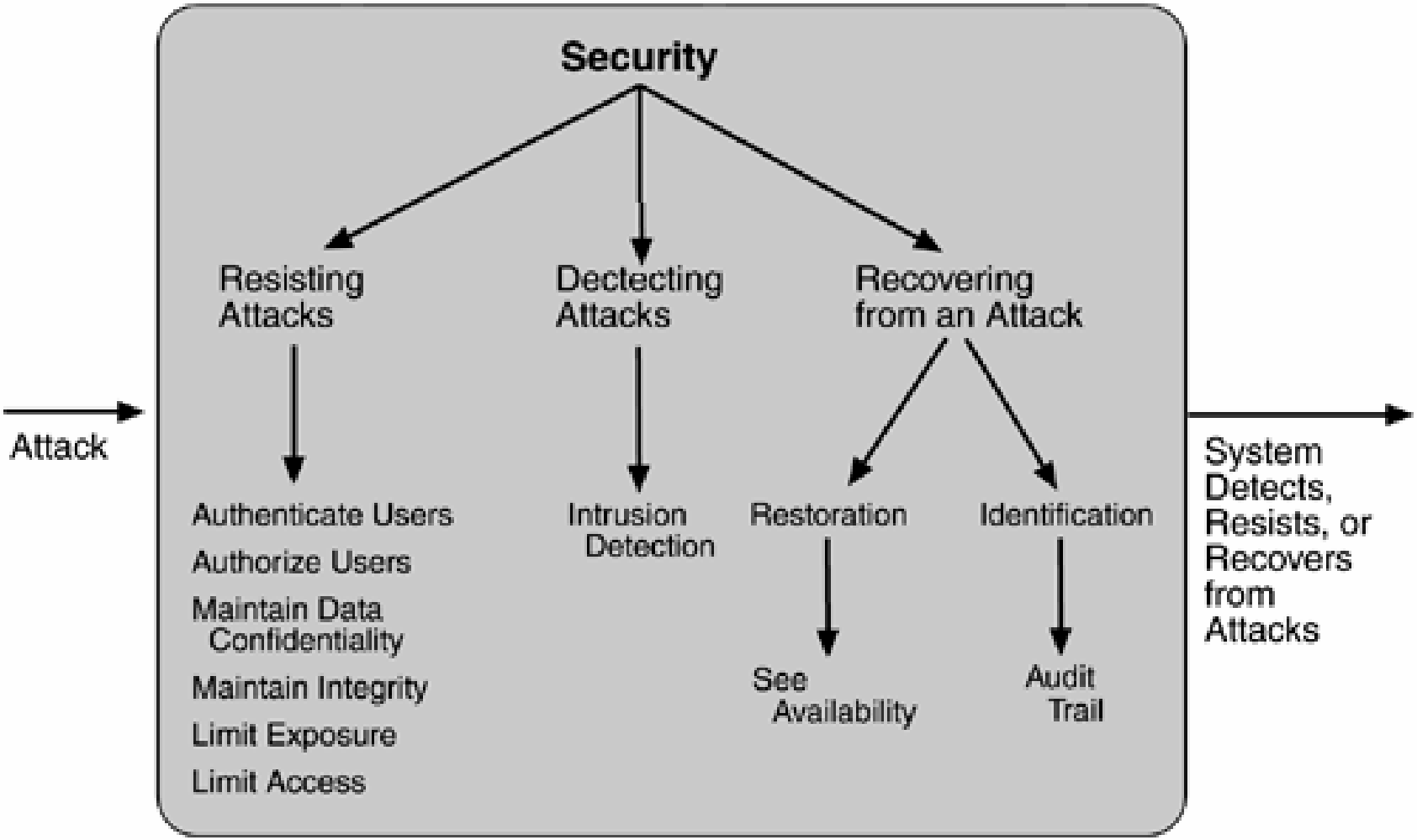
Security Tactics-recovering from attacks

- Tactics involved in recovering from an attack can be divided into
  - restoring state
  - attacker identification

- The tactics used in restoring the system or data to a correct state overlap with those used for availability
  - since they are both concerned with recovering a consistent state from an inconsistent state.
  - One difference is that special attention is paid to
    - maintaining redundant copies of system administrative data such as passwords, access control lists, domain name services, and user profile data.

- The tactic for identifying an attacker is
  - to maintain an audit trail.

Security Tactics-Summary



References

- Software.Architecture.In.Practice.2nd.Edition

Thank You!