CSci555: Advanced Operating Systems
Lecture 5 - September 23, 2005
Security

Dr. Dongho Kim
Dr. Tatyana Ryutov
University of Southern California
Information Sciences Institute

Security Goals

- **Confidentiality**
  - **inappropriate information is not disclosed**
- **Integrity**
  - **Authenticity of document**
  - **That it hasn't changed**
- **Availability**
  - **the ability of authorized entities to use the information or resource**

System Security: Terminology

- ***vulnerability*** **is a weakness in the system that might be exploited to cause loss or harm.**
- ***threat*** **is a potential violation of security**
- ***attack*** **is the actual attempt to violate security. It is the manifestation of the threat**
  - Interception
  - Modification
  - Disruption
- *security policy* defines what is and is not allowed
- *security mechanism* is a method or tool for enforcing security policy
  - Prevention
  - Detection
  - Reaction

Basic Security Services

Protection
Authentication
Access Control, Authorization
Accounting
Payment
Audit
Assurance
Privacy
Policy

Security Models

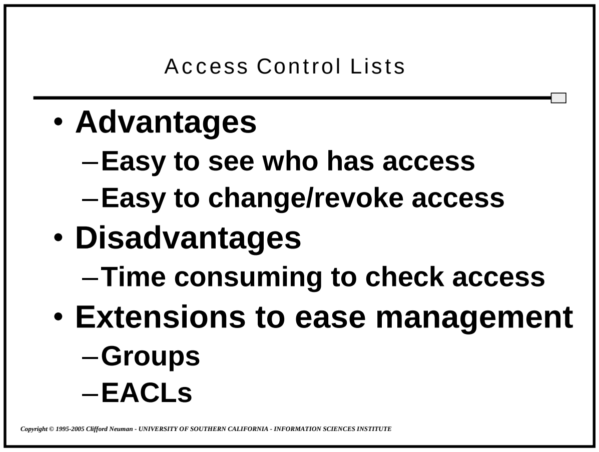
- **Discretionary Access Control**
  - **Users have complete control over his/her resources**
- **Mandatory Access Control**
  - **Administrators decide what you have access to as well as what you can give access to (as opposed to discretionary access control).**
  - **Users must deal with not having control over how they use their own resources.**

Security Policy

- **Access Matrix**

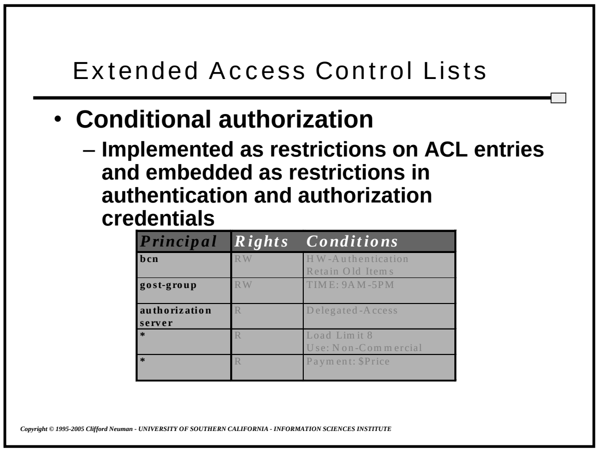
Subject	*O BJ1*	*O BJ2*
bcn	RW	R
gost-group	RW	-
obraczka	R	RW
tyao	R	R
Csci555	R	-

– **implemented as:**
  - ▪ **Capabilities or**
  - ▪ **Access Control list**



Access Control Lists

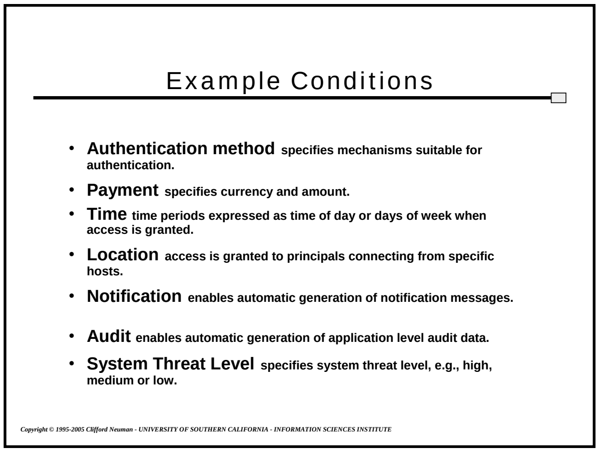
- Advantages
  - –Easy to see who has access
  - –Easy to change/revoke access
- Disadvantages
  - –Time consuming to check access
- Extensions to ease management
  - –Groups
  - –EACLs



Extended Access Control Lists

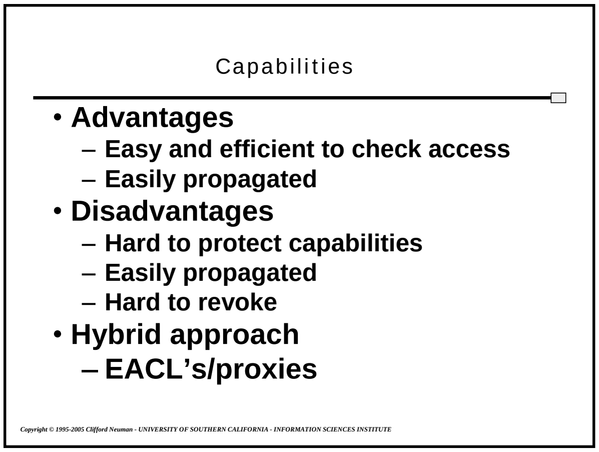
- Conditional authorization
  - – Implemented as restrictions on ACL entries and embedded as restrictions in authentication and authorization credentials

Principal	*Rights*	*Conditions*
bcn	RW	HW-Authentication Retain Old Items
gost-group	RW	TIME: 9AM-5PM
authorization server	R	Delegated-Access
*	R	Load Limit 8 Use: Non-Commercial
*	R	Payment: $Price



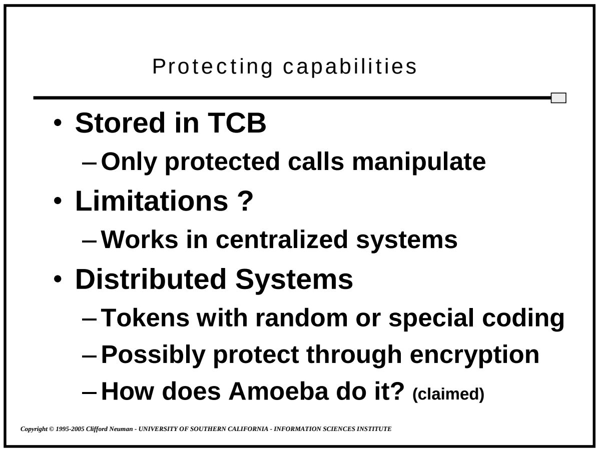
Example Conditions

- **Authentication method** specifies mechanisms suitable for authentication.
- **Payment** specifies currency and amount.
- **Time** time periods expressed as time of day or days of week when access is granted.
- **Location** access is granted to principals connecting from specific hosts.
- **Notification** enables automatic generation of notification messages.
- **Audit** enables automatic generation of application level audit data.
- **System Threat Level** specifies system threat level, e.g., high, medium or low.



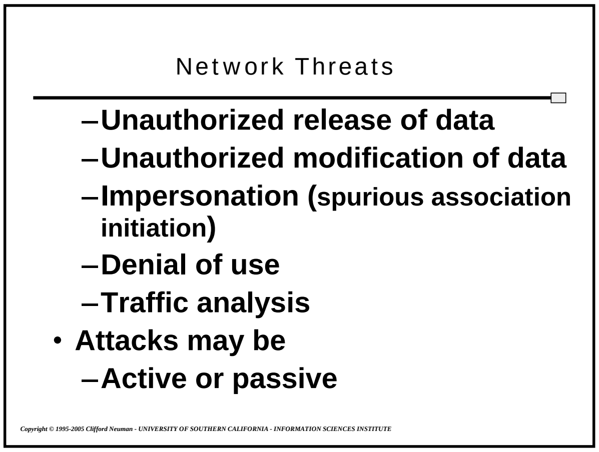
Capabilities

- Advantages
  - – Easy and efficient to check access
  - – Easily propagated
- Disadvantages
  - – Hard to protect capabilities
  - – Easily propagated
  - – Hard to revoke
- Hybrid approach
  - – EACL’s/proxies



Protecting capabilities

- Stored in TCB
  - –Only protected calls manipulate
- Limitations ?
  - –Works in centralized systems
- Distributed Systems
  - –Tokens with random or special coding
  - –Possibly protect through encryption
  - –How does Amoeba do it? (claimed)

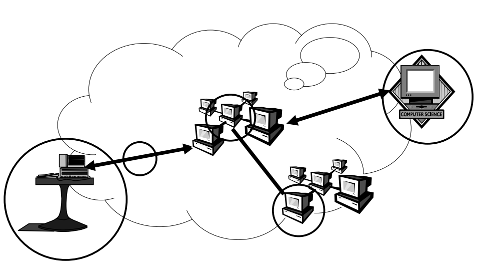


Network Threats

- –Unauthorized release of data
- –Unauthorized modification of data
- –Impersonation (spurious association initiation)
- –Denial of use
- –Traffic analysis

- Attacks may be
  - –Active or passive

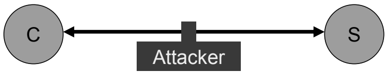
Likely points of attack (location)



Likely points of attack (module)

- Against the protocols
  - Sniffing for passwords and credit card numbers
  - Interception of data returned to user
  - Hijacking of connections
- Against the server
  - The commerce protocol is not the only way in
  - Once an attacker is in, all bets are off
- Against the client's system
  - You have little control over the client's system

Network Attacks



Eavesdropping

Listening for passwords or credit card numbers

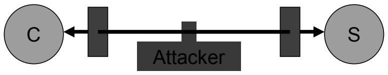
Message stream modification

Changing links and data returned by server

Hijacking

Killing client and taking over connection

Network Attack Countermeasures



Don't send anything important

Not everything needs to be protected

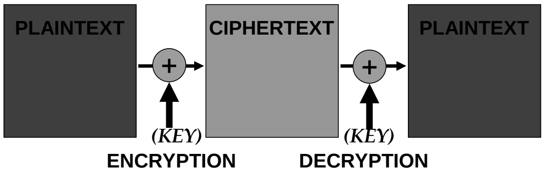
Encryption

For everything else

Mechanism limited by client side software

Encryption for confidentiality and integrity

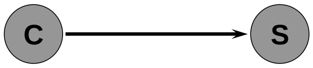
- Encryption used to scramble data

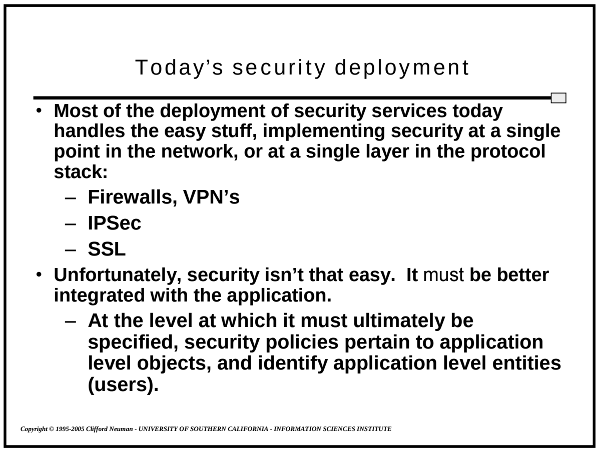


Authentication

- Proving knowledge of encryption key
  - Nonce = Non repeating value

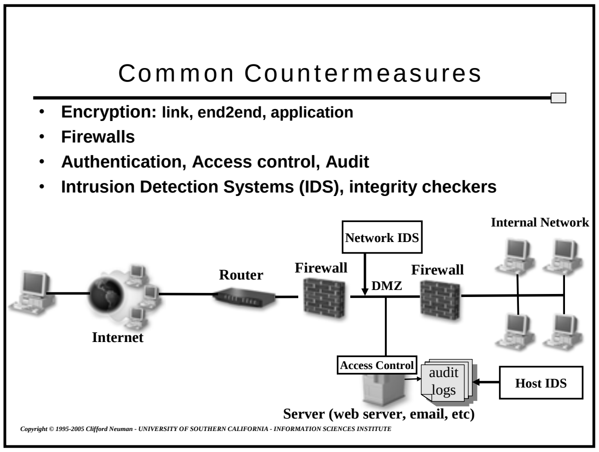
$\{\text{Nonce or timestamp}\}K_c$





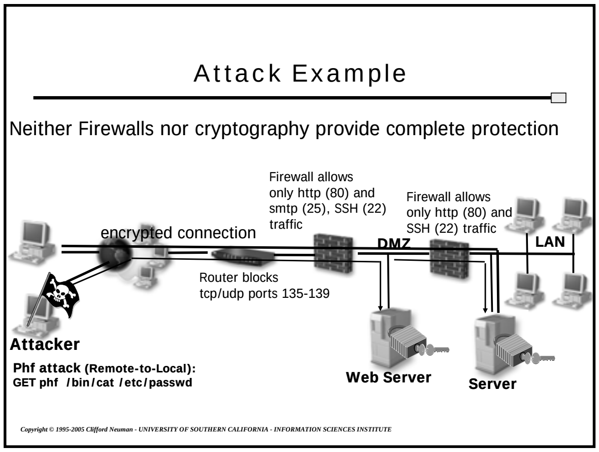
Today’s security deployment

- Most of the deployment of security services today handles the easy stuff, implementing security at a single point in the network, or at a single layer in the protocol stack:
  - Firewalls, VPN’s
  - IPSec
  - SSL
- Unfortunately, security isn’t that easy. It must be better integrated with the application.
  - At the level at which it must ultimately be specified, security policies pertain to application level objects, and identify application level entities (users).



Common Countermeasures

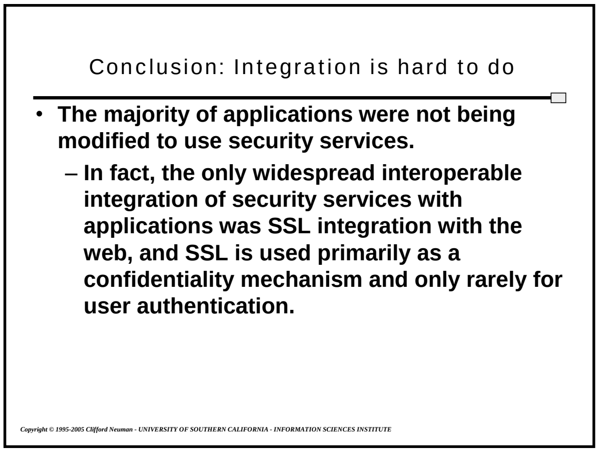
- Encryption: link, end2end, application
- Firewalls
- Authentication, Access control, Audit
- Intrusion Detection Systems (IDS), integrity checkers



Attack Example

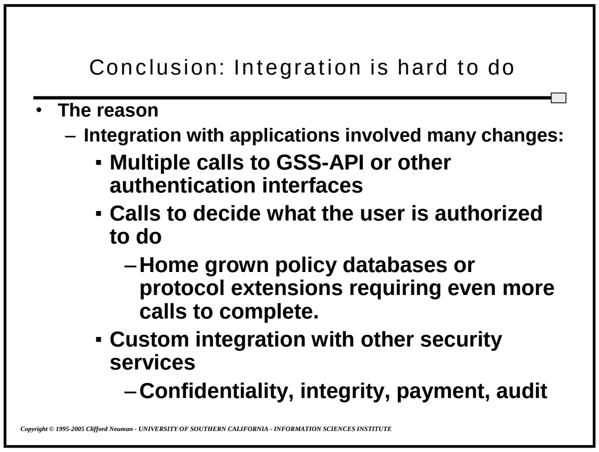
Neither Firewalls nor cryptography provide complete protection

Firewall allows only http (80) and smtp (25), SSH (22) traffic
Firewall allows only http (80) and SSH (22) traffic
encrypted connection
DMZ
LAN
Router blocks tcp/udp ports 135-139
Attacker
Phf attack (Remote-to-Local):
GET phf /bin/cat /etc/passwd
Web Server
Server



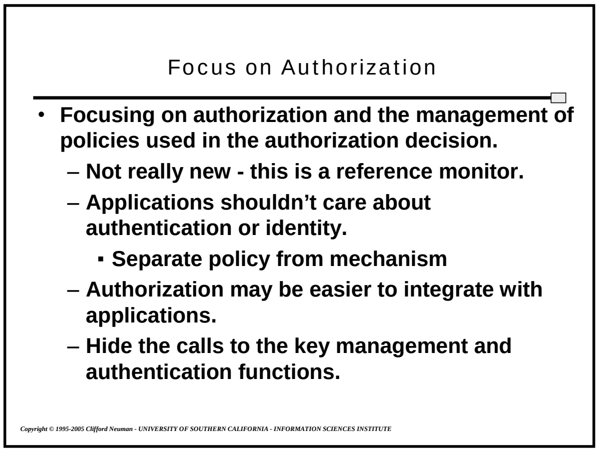
Conclusion: Integration is hard to do

- The majority of applications were not being modified to use security services.
  - In fact, the only widespread interoperable integration of security services with applications was SSL integration with the web, and SSL is used primarily as a confidentiality mechanism and only rarely for user authentication.



Conclusion: Integration is hard to do

- The reason
  - Integration with applications involved many changes:
    - Multiple calls to GSS-API or other authentication interfaces
    - Calls to decide what the user is authorized to do
      - Home grown policy databases or protocol extensions requiring even more calls to complete.
    - Custom integration with other security services
      - Confidentiality, integrity, payment, audit



Focus on Authorization

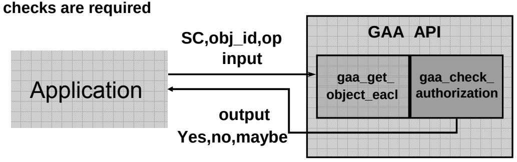
- Focusing on authorization and the management of policies used in the authorization decision.
  - Not really new - this is a reference monitor.
  - Applications shouldn’t care about authentication or identity.
    - Separate policy from mechanism
  - Authorization may be easier to integrate with applications.
  - Hide the calls to the key management and authentication functions.

Generic Authorization and Access-control API

Allows applications to use the security infrastructure to implement security policies.

gaa_get_object_eacl **function called before other GAA API routines which require a handle to object EACL to identify EACLs on which to operate. Can interpret existing policy databases.**

gaa_check_authorization **function tells application whether requested operation is authorized, or if additional application specific checks are required**



Credential transport (needed)

The GAA-API gets user & connection info from Security Context:

- **Evaluated and unevaluated credentials**
- **Delegated authority**
- **Cross-calls to transport to retrieve additional creds**

The security context is provided as:

- **Output from GSS-API (requires many calls)**
- **Credentials from transport or session protocols**
  - **SSL, ARDP**
  - **Other extensions are needed:**
    - **IPSec, pulled from Kernel, other extensions**

Integrating security services

The GAA-API calls must be made by applications.

- **This is a major undertaking, but one which must be done no matter how one chooses to do authorization.**

These calls are at the control points in the app

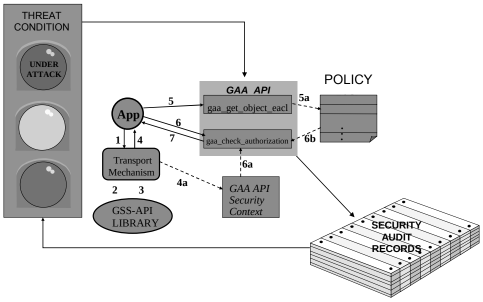
- **They occur at auditable events, and this is where records should be generated for ID systems**
- **They occur at the places where one needs to consider dynamic network threat conditions.**
- **Adaptive policies use such information from ID systems.**
- **They occur at the right point for billable events.**

Electronic commerce

Some authorization policies do not require user authentication at all - just that an item is paid for.

- **Policy specifies required payment.**
- **Cross call to credential transport retrieves payment credentials and grants access.**
- **If application used GAA-API, no change to the application is necessary, simply specify the payment policy instead of a more traditional identity based policy.**

ID and Audit relation to GAA-API



Application based ID

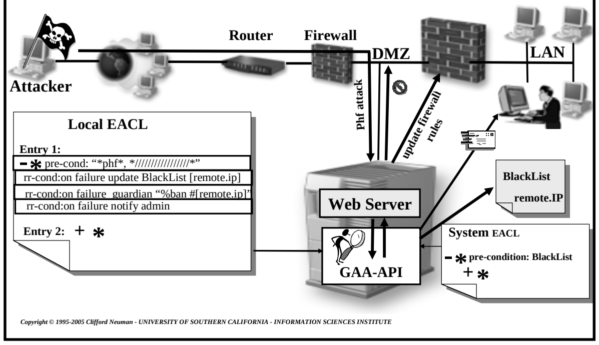
Without the GAA-API

- **Convince each application developer to add calls to audit functions in addition to all the other security calls they make (good luck). Of course it needs to do authentication too.**

With the GAA-API

- **Get developers to use the GAA for authorization decisions instead of making multiple calls to implement their own authorization database.**
- **Create module for GAA implementation that generates audit records according to policy.**
- **Write policy (inc. adaptive or credential based) that says when to generate audit records.**

Example 1: Web Server Exploit



Local EACL

Entry 1:

- ✱ pre-cond: “*phf*, *//////////////////*”

rr-cond:on failure update BlackList [remote.ip]

rr-cond:on failure guardian “%ban #[remote.ip]”

rr-cond:on failure notify admin

Entry 2: + ✱

System EACL

- ✱ pre-condition: BlackList

+ ✱