Sumedh

- Topics Covered

1. Protection
2. Access Matrix
3. Security
4. Prog, System & Net Threats
5. Cryptography & Encryption
6. Authentication
7. Firewalls & defence

- Protection

· Goals of protection

1. Principles of protect^n : Think of a computer as a collection of objects (h/w or s/w), like mmry files / devices

2. Unique access : each obj has a unique name and a defined way to be accessed

3. Protection Problem : The goal is to ensure that only permitted processes access object and do so correctly

· Principles of Protection

1. least privilege : Only give users, progs & system the min. permissions needed. This limits potential damage

2. Permission types : They can be static (set for process lifetime) or dyanamic (change when needed, like domain-switching / escalation)

3. Compartmentalization : Protect individual system components with specific restrictions

4. Granularity : Rough-grained, easier to manage but less secure like user either has / not has permissions
Fine-grained, more complex but more protection
eg ACL / RBAC

5. **Audit trail** : Records all access related actions to keep track

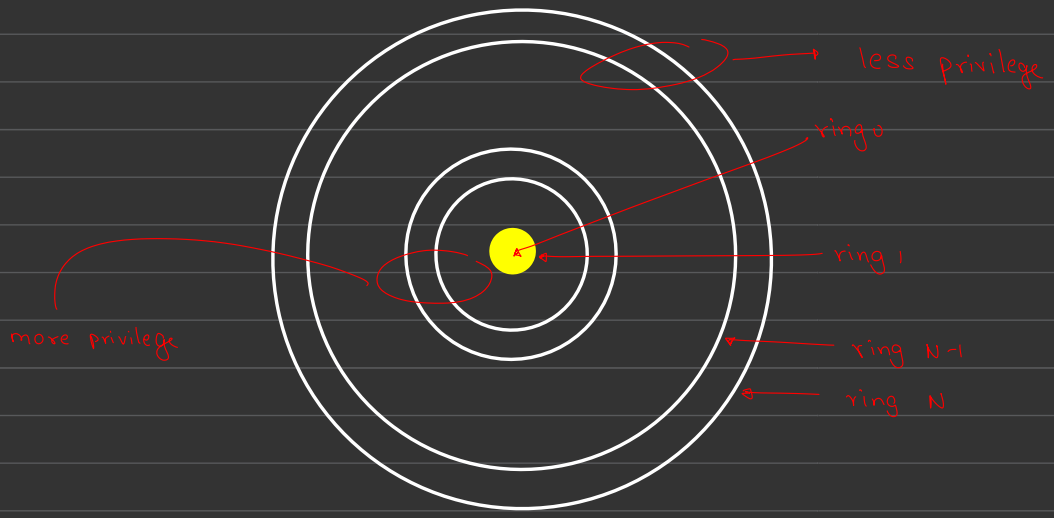
6. **Defence in depth** : No single prot. method solve all problms, we need multi-layer prot.

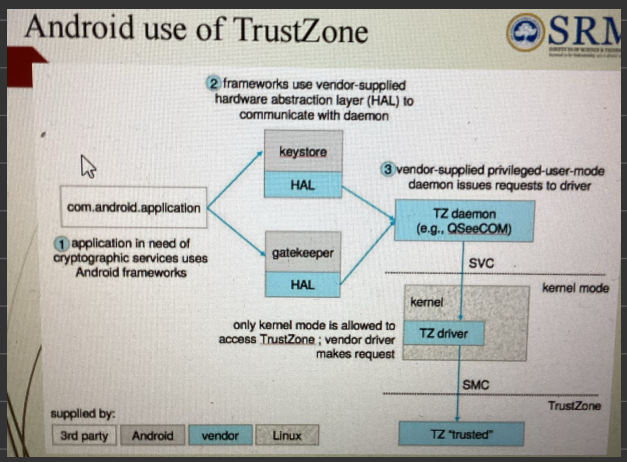
- **Protection Rings**

Privilege levels : System Components are organised by privilege level, often requiring hlw support

eg kernel operate in high-privilege ring while apps in lower ones

Similarly, **Hypervisors** & Adv CPU like **ARMv7** have a additional ring (Trust zone) for extra security layers, like protection sensitive data from kernel





- Domain of protection

1. Rings seperate functions into domains in a hierarchy, ensuring each process can only access what in needs

2. Processes shld also have access to resources of their current task

3. Process domain : A press operats within a domain specifying which resources it can access

4. Access rights r permissions defined by < Obj-name, right-sets > where right set is the set of allowed actions

5. In dyanamic system, processes can switch blw domains

- Domain Implementation

1. UNIX :-

- domain = user ID

- Domain switch

① via file system : ever domain a domain-bit (setuid bit) works when setuid = on

② via passwords : 'su' cmd allows a user to switch

③ via cmds : 'sudo' cmd allows switching

2. Android App IDs

- each app has a unique ID which acts as domain
- This id restricts access so others can't interfare
- Here domain protect resources & manage permissions

- Access Matrix

  └ Think of protection as a grid / matrix

Rows - domain (users)
Colums - objects (resources)

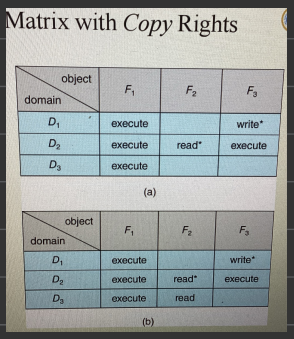
domain \ object	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

- Use of access matrix

1. Purpose : Controls what each domain can do with a object
2. User Control : The creator of an object can decide who can access it
3. Dyanamic Option : permissions can be changed dyanamically
4. Special right : Owner - full control
   Copy - Allows shares access
   Control - One domain can manage permissions
   Transfer - domain switch
5. Design : Mechanism (how it works)
   Policy (who gets access)
6. limitations : does not fully solve access problem

Access Matrix of Figure A with Domains as Objects

domain \ object	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

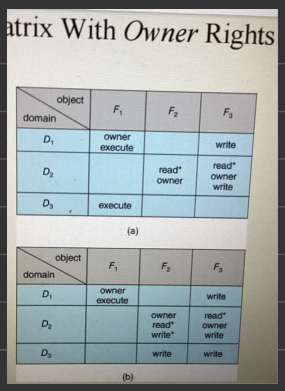

Matrix with *Copy* Rights

domain \ object	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute		

(a)

domain \ object	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute	read	

(b)

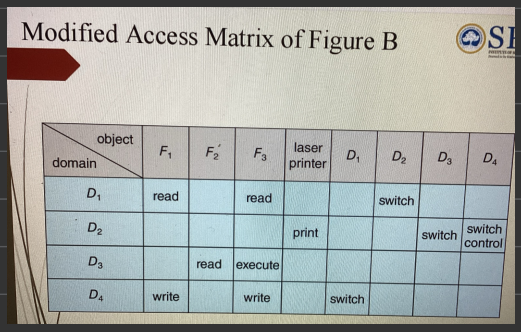

atrix With *Owner* Rights

domain \ object	F_1	F_2	F_3
D_1	owner execute		write
D_2		read* owner	read* owner write
D_3	execute		

(a)

domain \ object	F_1	F_2	F_3
D_1	owner execute		write
D_2		owner read* write*	read* owner write
D_3		write	write

(b)


Modified Access Matrix of Figure B

domain \ object	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch control
D_3		read	execute					
D_4	write		write		switch			

- Implementation of access matrix
  - L often large & complex

Option : 1 Global table

- list all domain / object permissions in one big table
- difficult to handle large obj

< domain , obj, right - set >

Option : 2 Access list of obj

- each obj has a domain with of obj rights
- can also have default set of rights

< domain , right - set >

Option : 3 Capability list for domains

- each domain has a list of obj of what it can do
- Capabilities like keys for specific permissions
- obj represented by name / add called a capability
- Capability list is associated with domain but never accessible to it

Option : 4 lock-key

- Compromises blw access & capability list
- Obj has locks & domain has keys
- Match is needed for access

- **Comparison of implementations**

1. Global table : Simple but large
2. Access list : User-friendly, but hard to mange access rights
3. Capability list : localizes access info but revocation is difficult
4. lock-key system : Flexible, easy share & revocation

- **Revocation of Access rights**

Diff methods to remove diff access rights

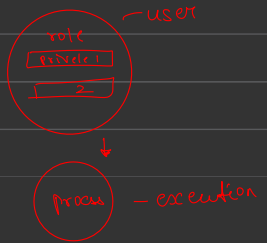
1. Immediate or delayed
2. Selective or general
3. Partial or Total
4. Temp or Permanent

Capability lists req location of capability to revoke

1. Reacquisation : Periodic reapproval
2. Back pointers : pts from obj to capb
3. Indirection : link caps to global table
4. Keys : Control acces

- **Role-based access Control (RBAC)**

- Used to apply permissions to more than just files
- Roles grant permsn & limit acss to certain calls
- Can give role via password



eg Oracle Solaris 10

- Mandatory Access Control (MAC)

Traditionally Discretionary Access Control (DAC) allowed user defined permissions but was weak

MAC is strong & can't be by-passed

labels : obj & user have labels to check access

Used in secured systems like SE Linux & TrustedBSD

- Other protection improvement methods

1. SIP (System integrity protection)

└ in mac OS 10.11

Slice up permissions

2. SIF (System call filtering)

like a firewall for system calls

3. Sandboxing

runs processes in a limited envo
has set of irremovable restrictions

- Security

- Security Problem

  - System Security : The goal is to make sure all resources are used only as intended

  - Intruders attempt
  - Threat is a potential security violation
  - Attacks can be accidental or intentional

- Security Violation Category

  1. Confidentiality breach (read private data)
  2. Integrity breach (tampering)
  3. Availability breach (destruction)
  4. Theft of service (unauth)
  5. Denial of service (Block servic)

- Violation Methods

  1. Authenticatn breach (Pretend to be smone)
  2. Replay attack (send mssgs)
  3. Man-in-the middle attack (comm blw 2)
  4. Session hijacking (bypass)
  5. Privilege escalation (gain high access)

- Safety Measure levels

Goal to have as much security as possible

4 levels : Physical : protect hlw
App : remove insecure apps
OS : buil-in OS security
Network : prevent attacks on net-traffic

- **Program threats**

  └ many variations , many names

1. **Trojan Horse** : Malicious code hidden inside legitimate software which can spy

2. **Trap door** : Code that bypasses security checks

3. **Spyware & spam** : Spyware can lead to spam, making infected comp display junk mssgs

```c
#include <stdio.h>
#define BUFFER SIZE 256
int main(int argc, char *argv[])
{
   char buffer[BUFFER SIZE];
   if (argc < 2)
    return -1;
   else {
    strcpy(buffer,argv[1]);
    return 0;
   }
}
```

→ can check logic errors or flaws

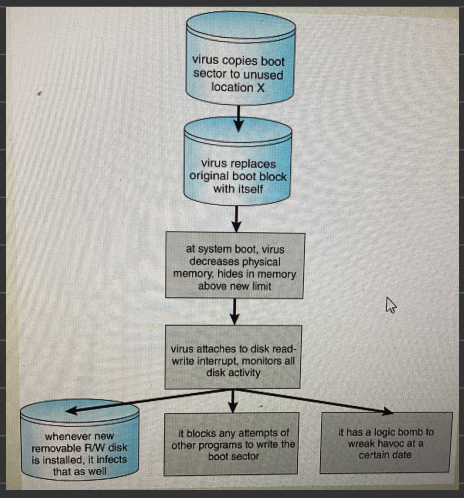
- **Viruses**

  - It's a code fragment in a legit prog
  - Is self-replicating
  - designed to infect other computers
  - often spread thru emails

Virus dropper : They insert virus into the system

Types of viruses :- File / parasitic
Boot / mmry
macro
Stealth
Encrypted
Armored

How a virus attacks →



- evolution of attacks : Attacks started as escps but are not used in organised crime

- System & Network Threats
  - some systems are open by defaut so easier to attack
  - Hard to protect network connection, specially with no phy barrier once connected
  - Also identifying the threat is hard as only IP add known

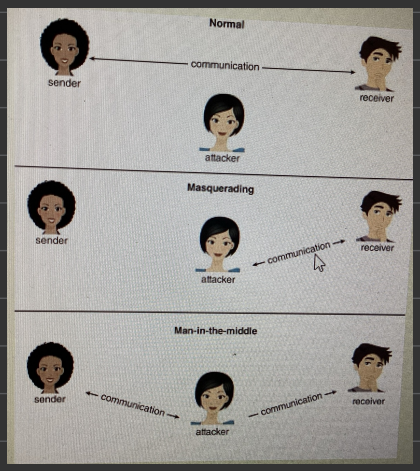
- worms
  - Standalone progs
  - They replicate across systems
  - Eg are Internet worm in UNIX

* Grappling hook uploaded main worm prog

- Port Scanning
  - Scanning IP add to scan open ports
  - Here attackers use to detect protocols / os
  - Tools like nmap & nessus help attackers identify vulner.
  - Zombie system used to mask the real location

- DOS (denial of service)
  - Overload the target comp by preventing it to do other work
  - Come from multiple site at once
  - Makes traffic to a website
  - Used for good & evil

(basically bullying)



- Cryptography & Encryption

- Crypto as a security tool
  - Crypto is a versatile tool
  - On a single comp, mssgs are manged by OS
  - While mssg on net & their sources can't be trusted without cryptg

Purpose : limits who can sent & receive mssgs

  - Secrets & key are central to cryptography

Benefits : Confirmation of source
Builds Trust

- Encryption

Purpose : Controls who can read a mssg

Components : Keys (S) : Secret codes
Mssg (M) : Og data
Ciphertext (C) : Encrypted data

Encryption function : Use keys to turn mssg into ciphertext

A func E : K (M → C), for each k ∈ K, E_k

decryption function : vice-versa

A func D : K (C → M) for each k ∈ K, D_k

Two Types of encryption

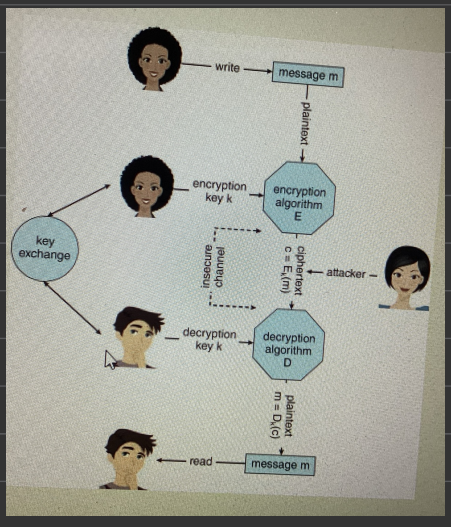
1. Symmetric
2. Asym

- Symmetric Encryption

- Uses same key encrypts & decrypts
- The key stays a secret

Algorithm

1. DES : older encryption method
2. Triple-DES : More secure than DES
3. AES : Modern & secure
4. RC4 : Is weak



- Asymmetric

Two keys : public key (encrypt)
private (decrypt)

eg RSA : common type of RSA encryption

- The public key can be shared while private key must be private

RSA eg : Uses two large prime no's to create keys

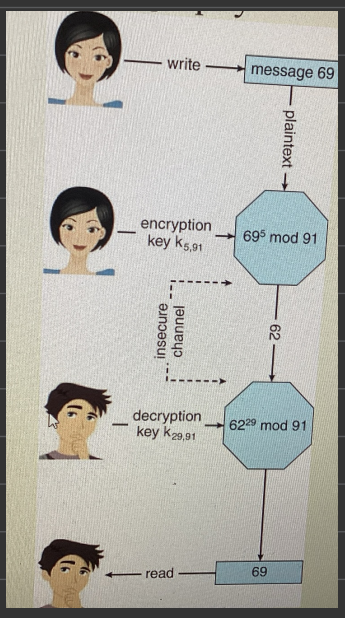
Math behind : encrypt by a math relation & cannot be reversed

eg 7 & 13
p q

→ $N = 7 \times 13 = 91$ $(p-1)(q-1) = 72$ } learn

ke prime to 72

∴ $k_e = 5.91$
$k_d = 29.91$



- **Authentication**

Goal : limit to send a mssg

Work with encryption : Proves a mssg is unaltered

Components :
- K : set of keys
- M : " " mssgs
- A : set of auth
- S func : Creates auth for mssgs
- V " : Check if mssg valid

$S : K(M \rightarrow A)$ where $k \in K$, S_k

$V : K(M \times A \rightarrow \{true, false\})$, where $k \in K$ V_k

How it works :
- Comp with key K can create mssg
- Comp w/o " can't " " or fake
- Auth are sent with mssg w/o reveal

- **Hash functions**

Purpose : makes a small, fixed-size "digest" of a mssg

Collision resistant : Hard to find two diff mssg with same hash

Eg : MD5 & SHA-1

limitation : If the hash function known, someone can alter it

$H(m) : H(m')$, the $m : m'$ (learn)

- **MAC (Mssg - auth code)**

Purpose : uses symm encrp to create a "checksum" or small code auth

Security : Only ppl with sec key can create & verify code

benefit : Can securely check long mssgs

- **Digital Signatures**

Purpose : Use Asym encryp to create digi signs

Public key verifies
Private key signs

eg RSA

Adv : Anyone can check auth

- **User Auth**

Purpose : Verify a user's identity on user ID

Password basics : must be secret
" " non guessable

Security : log invalid attempts
Encrypt passw

- **Passwords**

1. Encrypt : To keep them safe
2. Salt : Add salt to ensure unidentity
3. One-time passw
4. Biometric : via fingerprints

- Firewalls & Defence

  - Implementing security defence

    Defence in depth : use multiple layers

    Components : Security Policy
    Virus prot
    Safe computing
    Intrusion detection

  - Firewalls

    Network firewalls : Create a boundary blw trusted & untrusted network

    Prevent unauth access

    Types : App proxy firewalls
    System call firewalls

    limitation : They can be bypassed

  - Comp system Classification

    D : Minimal sec
    C : Basic protection
    B : Enhanced sec
    A : High lvl sec

eg Windows 10 security Model

- each user has unique security ID
- Upon login, token grant permissions
- System tracks permission
- Obj have security