IPV4 (32-bits) → 2^32 IP addresses.
IPV6 (128-bits)

Foundation for Advancement of Science & Technology

Lecture - 14

→ Classification of Malware

- Mem Based: (Resident virus)

TSR → a virus which attaches itself with the main running program even after the program is terminated. The virus resides in memory (RAM)

- Temporary Mem Virus
- Swapping Mem Virus
- Non-Resident (Harddisk)

Search Function

Copy Function

morph → shape/form
mono → one
poly → many
virus signature (virus definition)

- OS
  - user process
  - kernel process (high privileges after infection)
    - close to user
    - viruses hook themselves in kernel through a system driver like programs.

→ Obfuscation Techniques

↳ to hide

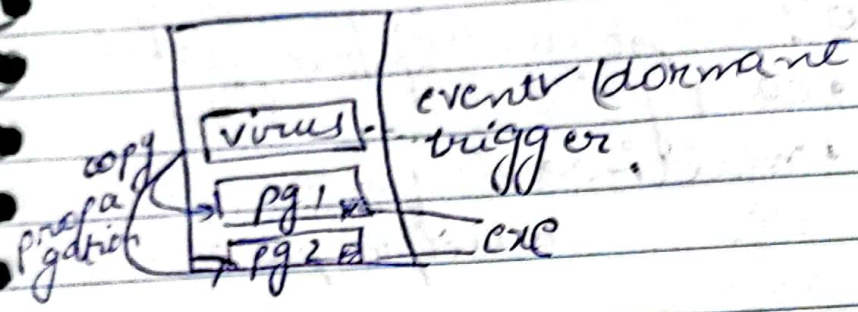
Oligopoly → Two or Three

Monopoly → One company in the industry to provide specific product/service

benign → harmless
malicious → harmful

Backdoor function → user do not have access to it, only the maker/developer knows.

→ Phases of a viruses (4):
- Dormant → idle, waiting for some condition
- Trigger → started to infect / activated
- Propagation → copy itself into other programs
- execution → execute the payload.



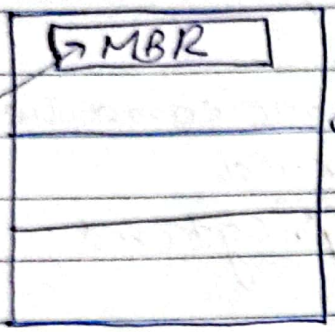
payload → actual virus code (executed)

types of viruses → virus, worm, trojan

→ Trojan / Trojan Horse
↳ non-replicating program jo dekhne mein harmless ha but asl mein harmful ha.

→ Worm: (eg SQL Slammer → UDP port SQL) → worm of 376-byte
- self replicating program. Without any user interference, send copies of itself using network. eg using e-mail
- has phases like viruses. (4)
- can cause enormous damage.

→ ~~Fff~~ ~~files~~ ~~that~~ Things that are infected
- Executable
- Interpreted File
- Kernel
- Service
- MBR (Master Boot Record)
- Hypervisor



→ Where does the code go?
- Overwriting
- prepend / Append
  ↳ did not modify code
- cavity → overwriting unused areas of executable files
- multi-cavity → cavity on multiple places.

→ Packers
↳ compresses or encrypts data

→ Malware Analysis:
Process of determining the functionality, origin and potential impact of a given malware.

Dead Code → The code is not executed.

IDA → Interactive Disassembler

PEiD → Portable Executable

Foundation for Advancement of Science & Technology

Lecture - 15

Malware Analysis:

- Computer Security incident management
- Malware Research
- Indicators of Compromise (IOCs) extraction.

Use cases

→ Types — Static → analysis in which code is not run. (code analysis)
dead code is read & understand.

Dynamic → conducted by observing & manipulating malware as it run
Needs a safe environment (Sand Box)

- Static analysis is safer then the dynamic analysis because the code is not being run

→ Static is slower and safer as compared to dynamic. Moreover, static doesn't require sand box whereas dynamic does.

→ Incident Handling.

The attack handling process

- Preperation → Get the team ready.
- Identification → Identify if event is incident
- Containment → Limit the spreading
- Eradication → Removal
- Recovery → Restoration
- Lessons Learned → Be prepared for next time.

IDS → intrusion Detection System.

→ Malware Defenses:
- Detection
- Removal
- Identification

→ software security (Many other)

→ Control Hijacking (somewhat related to the use of C, the way C manages memory, and system calls like exec())
-) buffer overflow, integer overflow
-) Goal: Take over target machines

Memory
- static → global variables, or static variables
- stack → variables within function
- heap → dynamically allocated memory.

heap: non-contiguous, somewhat unlimited, large

stack: contiguous, limited (LIFO).

→ new function is executed
↳ "stack frame"

SP → Stack Pointer

can skip JavaScript Part

• NOP → No Operation

• void * → enable karta hai runtime pe iss ko kisi array ki ref point karwa dein.

• func * → enables karta hai runtime pe iss ko kisi function ki ref point karwa dein.

Foundation for Advancement of Science & Technology

Lecture-16

→ Counter Measures

- Buffer overflow
- Integer overflow
- format string vulnerabilities

→ nop : wasting an instruction cycle.

→ Attacker should ~~not~~ consider:

- The program should not contain '\0' character (null)
- Overflow should not crash program before func() exits.

→ ~~Buffer overflow~~

→ virtual tables:

virtual table.

→ Fuzzing: → method to find overflow

Test cases → exhaustive testing for all possible inputs.

trying all types of input values (may be automated)

IP address 255.255.255.255 → reserved for broadcasting, cannot be assigned to any machine.

→ Integer Overflows:
When overflow happens, compiler doesn't generate errors. It uses techniques to handle it
- modulo
- truncation

For example, short int is used but the value of input is greater than the size of short int i.e 16-bits

For ~~default~~ signed int (32-bits)	For signed int (default)
shortest value → 0	neg → $-(2^{31})$ ~ 0
largest value → $2^{32}-1$	pos → 0 ~ $2^{31}-1$
	(MSB is used for storing signive tive or -ive)

- buffer overflow can lead to integer overflow.

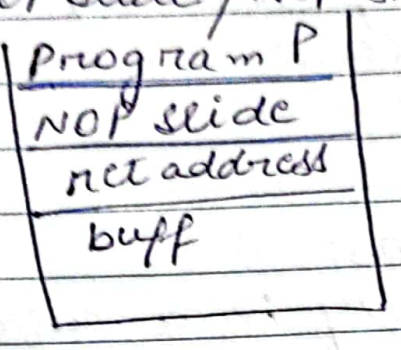
→ Format String Problem
Eg:
printf("%s hello!\n", buf)
buf: user input pointing to specific address location

- There is a chance of integer overflow in case of type-casting.

Lecture - 17

→ How does attacker determine the ret-address?

Solution: NOP slide / NOP sled



→ Format String Attack:

~~printf(fp~~

→ Database Security:

↓

Structured collection of data used by one or more application

- DDL → data definition language
- DML → data manipulation language

→ Authorization Tables: Access Security

→ flat file database:

inserting data in 1 file. NO table. NO unique identifiers etc.

→ SQL injection attacks:

↳ sends malicious SQL commands to the database server.

• SQLi → SQL injection

SQL injection can also be exploited to
- DoS attacks (Denial of Service)
- Modify or delete data
- arbitrary

→ Injection Attack Steps:
- find a vulnerability in a custom Web app
- Web server receives malicious code
- - - -

→ The SQLi attack typically works by prematurely terminating a text string and appending a new command.

Example of SQL injection

→ In-band attacks:
i) Tautology
ii) End-of-line commands
iii) Piggybacked queries

i) injects code in one or more conditions so it evaluates to true

ii) legitimate code that follows are nullified through use of end-line comments after injecting code.
iii) attacker adds the additional queries.

→ inferential attack:

No transfer of data, but attacker is able to reconstruct information by sending particular requests & observing the resulting behaviour of the website / database server.

↳ illegal / logically incorrect queries

↳ Blind SQL injection

↳ Attacker asks questions from server to observe functionality in every case.

→ SQLi countermeasure

- Defensive coding
- Detection — Signature based → match specific attack patte
  - code analysis
    - ↳ use of test suit to detect SQLi vulnerabilities.
  - anomaly based → attempts to define normal behaviour then detect pattern that do not match
- Runtime Prevention

DB security → CH:05
(Stalling. CS: P+P)

Foundation for Advancement of Science & Technology

Lecture - 18

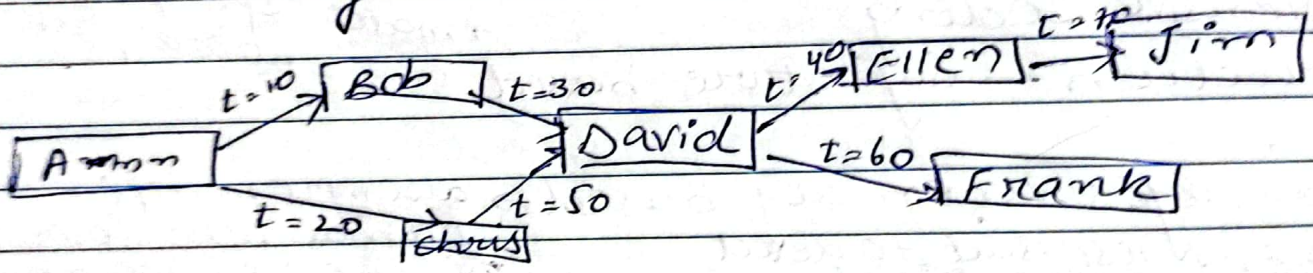
→ Access control in DB Systems

'GRANT' command → provides access to DB user for a specific DB object

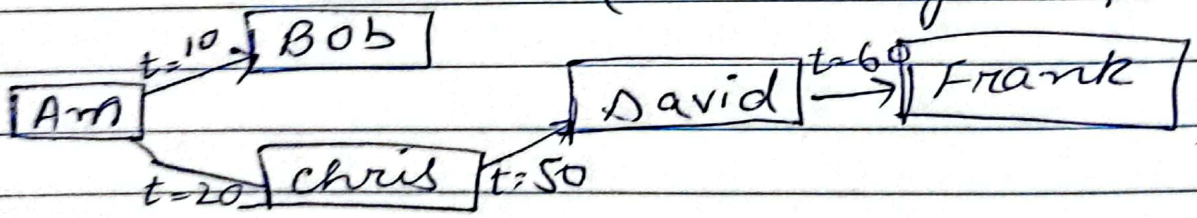
"WITH GRANT OPTION" → allows to give grant to other people rights to "GRANT" others.

"REVOKE" → remove access rights

→ cascading Authorizations



↓ ('REVOKE' rights of David)



→ Role-Based Access Control

Oracle
IBM DB2
MS SQL Server
} Proprietary DBMS

SQLite
MySQL
PostgreSQL
} open-source

→ Output Perturbation techniques — Random-sample query
Statistic adjustment

- DB user categories:
  - Application owner
  - End-user
  - Administrator

→ Inference Attacks:
↳ performing authorized queries and deducing un-authorized info from the legitimate responses received.

→ Counter measures:
- Detection at database design
- Detection at query time.
  ↳ not easy to manage

→ Ex of inference

→ Statistical Databases: (provides data of statistical nature)
Types:
↳ Pure Statistical Database
Only stores statistical data
↳ Ordinary DB with stats access
combo of common queries & stats data

→ SDB: Characteristic Formula
- OR → '+'
- AND → '*'
- NOT → "—"

Solution:
- SQL Restriction
- Perturbation (Adding noise)

→ Database Encryption
- can encrypt entire database
- " " rows / columns
- " individual entries / fields.

→ Issues
- Lacks flexibility
- No obvious way to do this
- Do not preserve ordering of values.

SOP: Same Origin Policy | Cookies → Session Management
AJAX → asynchronous javascript
↓
Ex: refreshes Gmail itself to perform an ac show latest mail without the need for user to perform an action

Foundation for Advancement of Science & Technology

Lecture-19

Cross Site Request Forgery (CSRF)

→ CSRF Attacks
→ Counter-measures

→ How are user sessions managed on the Web?
→ AJAX, Javascript, iframe

→ Web Security:
- Website → collection of webpages (objects)
  - HTML, CSS
  - Javascript
- Https/Http → Request / Reply
- Webserver → Stateless
- Cookies → used for Session Management

• CSRF: → attack
forces end-user to execute un-wanted actions

• Impacts
  • Limited to the capabilities exposed by vulnerable application

→ CSRF counter-measures
• Tokens (STP)
  ↳ Synchronization (Secret) Token Validation
•

→ CSRF Defenses: • Origin Header • Referer Header

Foundation for Advancement of Science & Technology

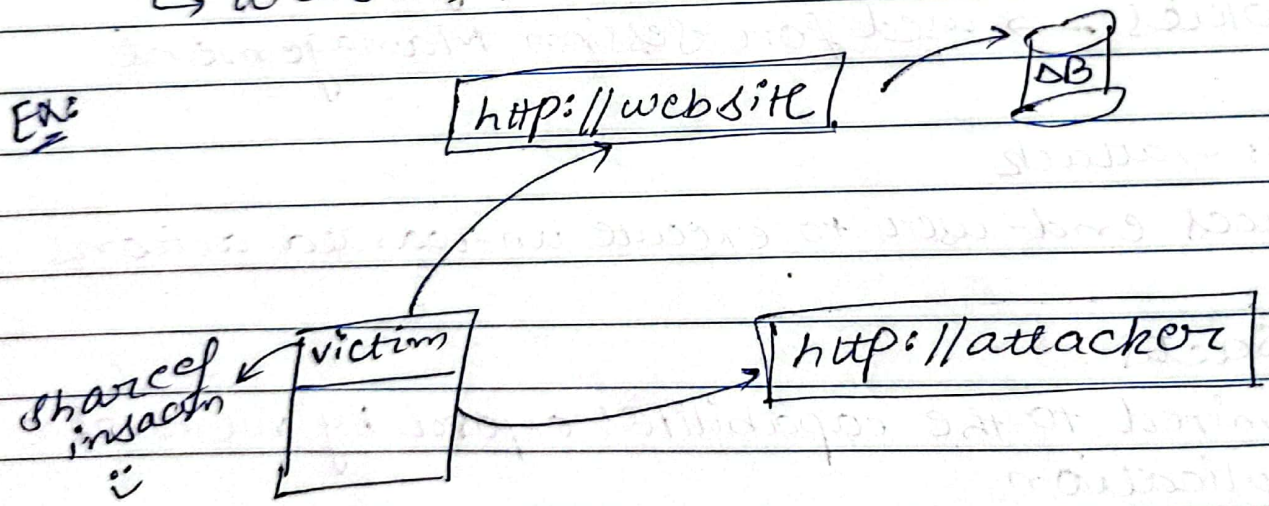
Lecture - 20

(previous) CSRF → cross site req forgery
XSS → cross site scripting
→ countermeasures
→ authentication

→ Origin: (Same - Origin Policy)
↳ protocol
↳ domain name
↳ port

→ XSS: (Cross site scripting)
↳ website, victim and attacker

Ex:

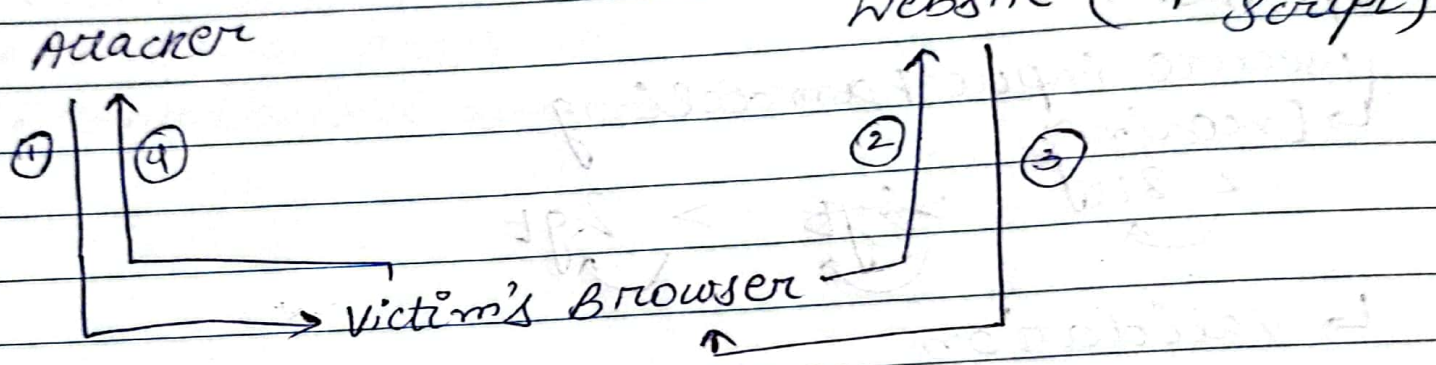


→ XSS attack can do:
- cookie theft
- key logging
- Phishing
- malicious Java Script is executed in the context of that website.

HTML code → document.form[0]
↳ allows you to get all the properties relevant to form[0]

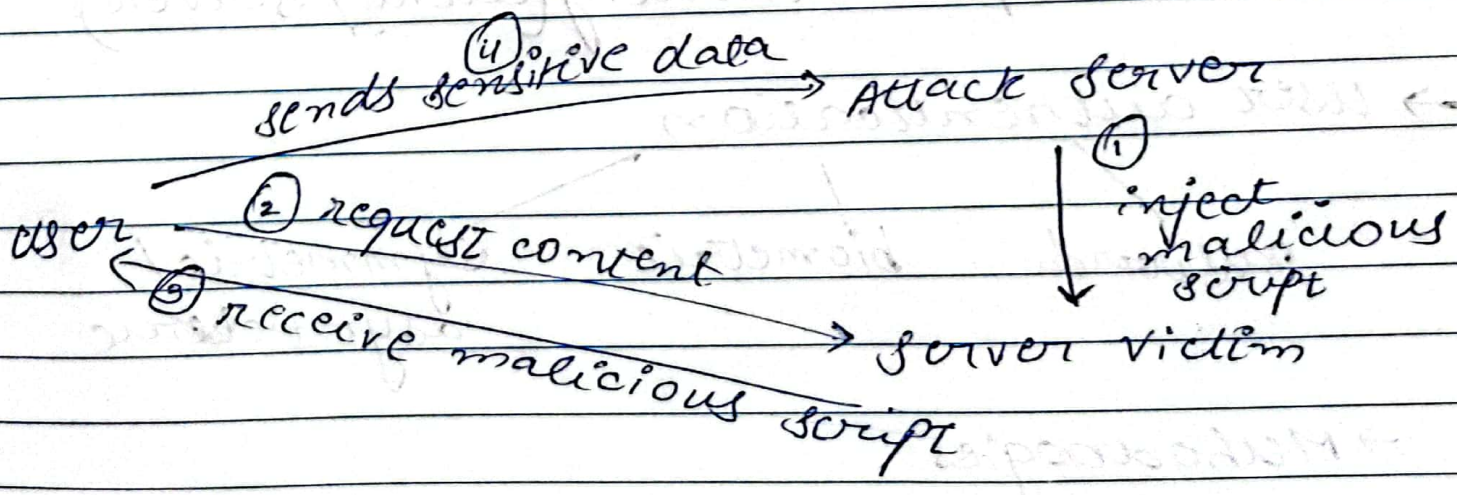
→ Types of XSS — DOM-based
Reflected
Stored

• Reflected Ex:



• Stored XSS
↳ perpetrator has to locate a vulnerability in a web application & then inject malicious script

Ex:



Foundation for Advancement of Science & Technology

- DOM
  - ↳ document object model
  - ↳ attacker tries to modify DOM

→ XSS more dangerous than CSRF.

→ Defenses:

- secure input handeling
  - ↳ Encoding
    - < &lt
    - ~~>~~ &gt
    - > &gt
  - ↳ validation
    - classification strategy
    - Validation Outcome
      - sanitized → modify

↳ Input Handling contexts
↳ Secure Input Handling (client / Server)

→ User authentication

- password
- biometric
- symmetric / asymmetric

→ Methodologies

- you know
- you have
- you are

Foundation for Advancement of Science & Technology

Types of Authentication

- Repudiable Authentication
- Non-Repudiable Authentication

→ Authentication Mechanisms

- Basic authentication involving a server
- Challenge-response
- Centeralized authentication

Foundation for Advancement of Science & Technology

Lecture - 21

User Authentication:

→ Password Auth
→ Biometric Auth
→ Secure ID
→ Kerberos & KDC
  → TGS → Tickets

- Repudiable Auth → password
- Non-Repudiable Auth → Biometric (fingerprint)

→ Password Auth

/ user create & transmit proof (client)

\ verify the proof (server)

(UserID, passwor) = ~~HUI~~ hash(password) = HUID

→ "salt"
  ↳ a value, random number which can be used to perform several mathematical operations

H(pswd, salt)
  ↳ hash product

(continuation in pp section)

Information Security

Lecture-21 (continue)

→ Using "salt" & hash:
Compute HP = hash(password | salt) → usually given by system.
Store the triples {UID, HP, $salt_{UID}$}

- Even if 2 users ~~enter~~ sets the same password but because the salt provided will be different, so the hash product/value for both the users will also be different.

→ Problems with Biometric
- FAR = False Acceptance Rate
- FRR = False Rejection Rate

→ System challenge-response Authentication:
- "NONCE"
  Number One Number used Once

Ex: Recaptcha

→ Asymmetric challenge-response Auth:
- public & private keys are also used.

→ Secure ID: Architecture → Invented & patented by security dynamics.
→ Time-based synchronus OTP ~~architecture~~ technique
- $P_{UID}(t) = h(S_{UID}, t)$

ally given
system.

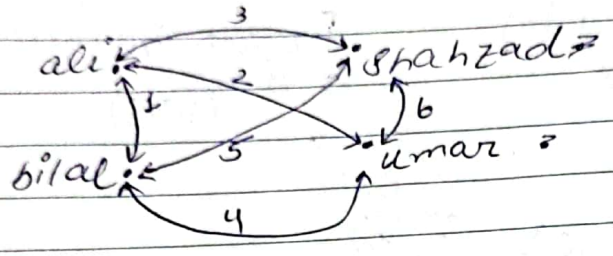
password
ifferent,
the

ion:

ed
mics.
e

→ Secret keys for N-system Network:

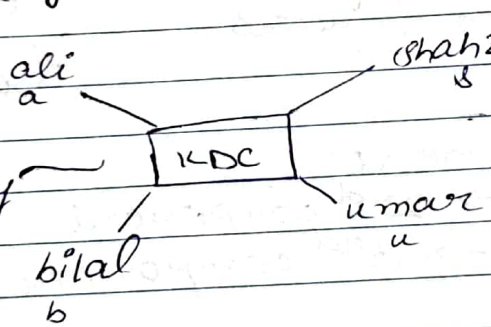
Ex:



N = 4 (No. of Nodes/Us

pair of secret key = $\frac{n(n-1)}{2} = \frac{4(4-1)}{2} = \frac{12}{2} = 6$

→ KDC (Key Distribution Center) → has all the keys



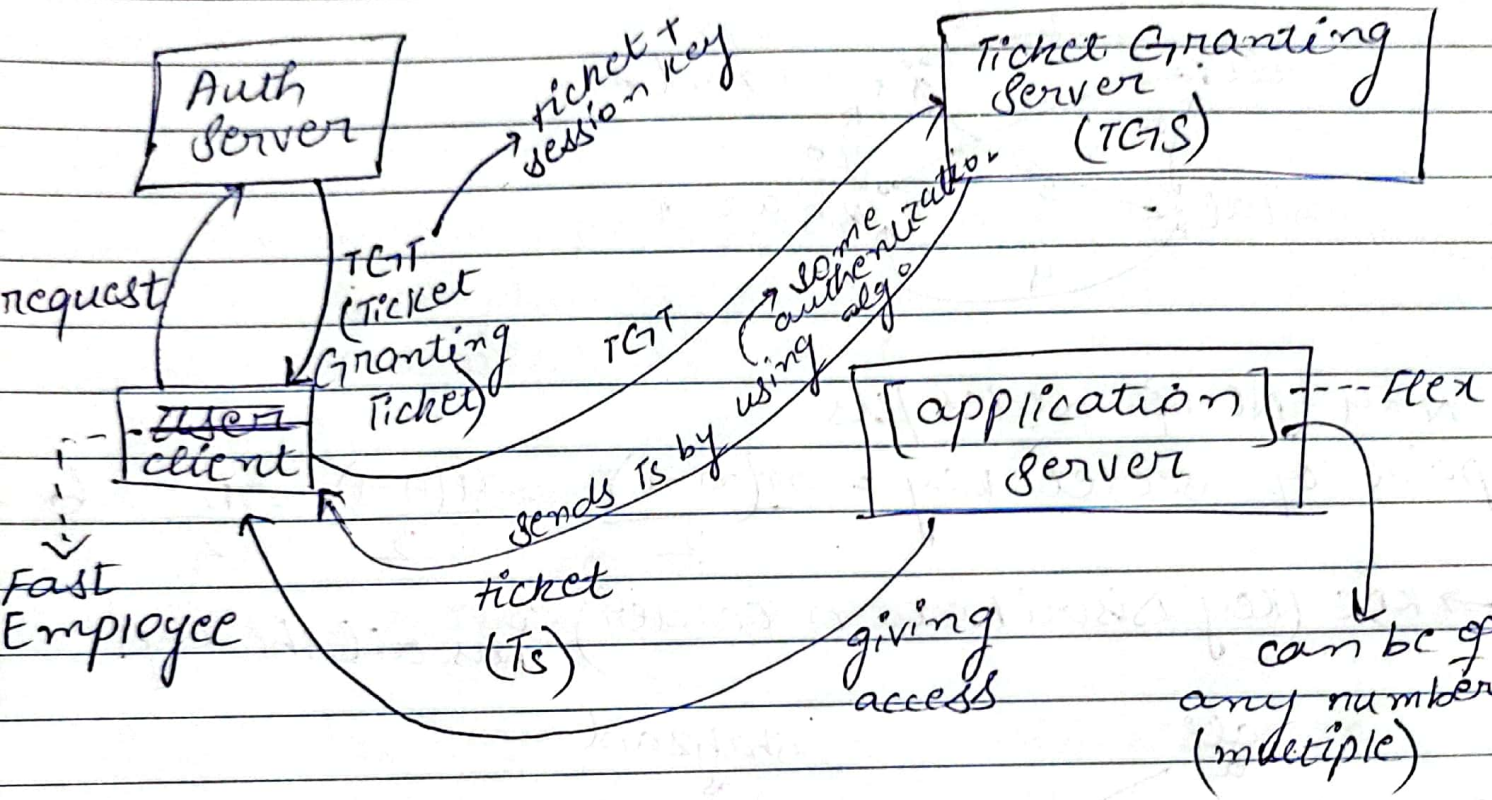
ali ↔ bilal
KAB ~ A's key, B's key

- KDC → a single point of failure
- KDC → always online (for offline → need a machine to store backup

→ Kerberos:

- Protect against eavsdropping & firewall limitation to users & replay attacks.

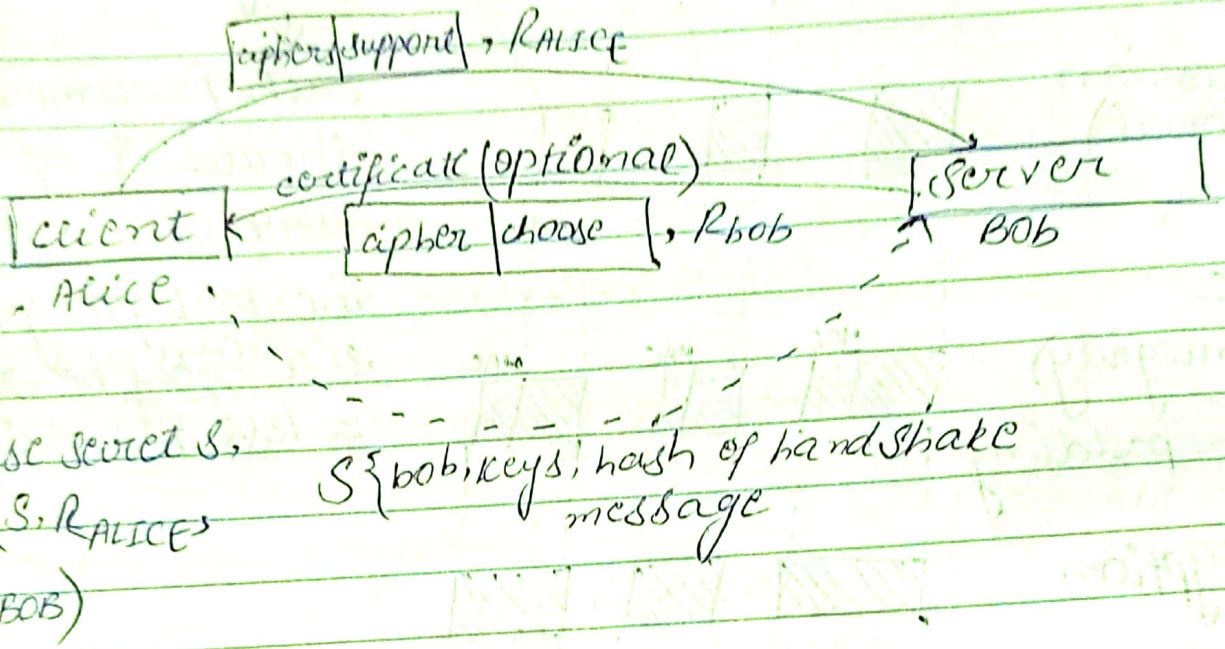
→ Kerberos overview



- client → 1st component
- Auth Server, TGS → 2nd component
- [application] server → 3rd component

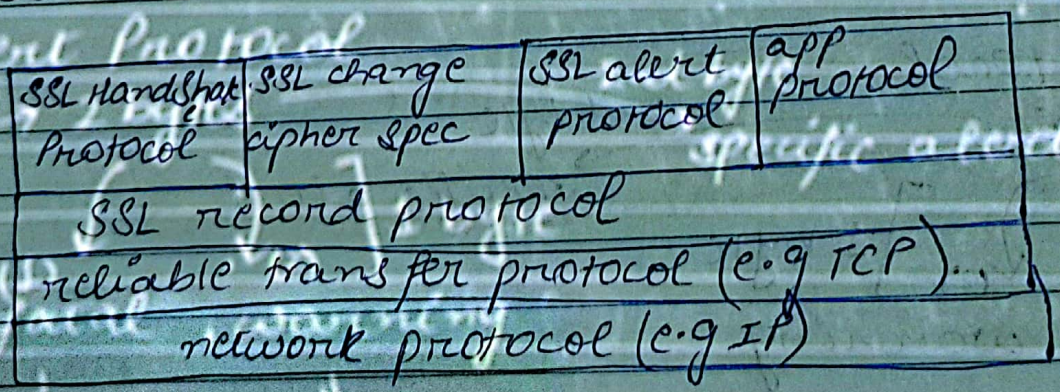
Foundation for Advancement of Science & Technology

•) Simplified SSL - Handshake:

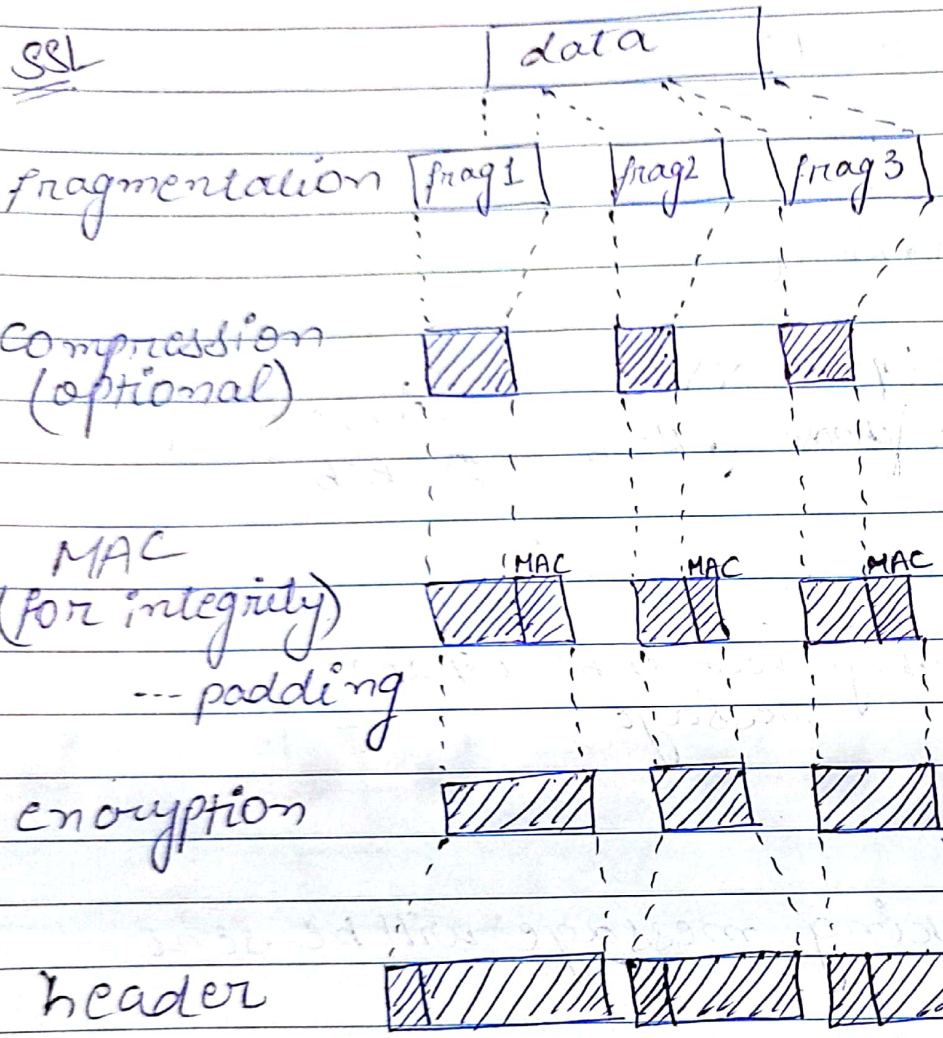


•) After ~~se~~ handshaking message will be sent.

→ SSL : V3 Architecture



SSL



→ 2^{14} bytes

↓

each fragment should be of same size (2^{14}).

The last fragment size can be ~~may be~~ ~~=2^14~~ ← less than 2^{14}

→ Alert Protocol

↳ 2 bytes

2nd byte → specific alert

1 byte

(1) fatal

(2) warning

↓ something very serious

↓ not that serious

Foundation for Advancement of Science & Technology

ByteByteGo
↳ youtube channel

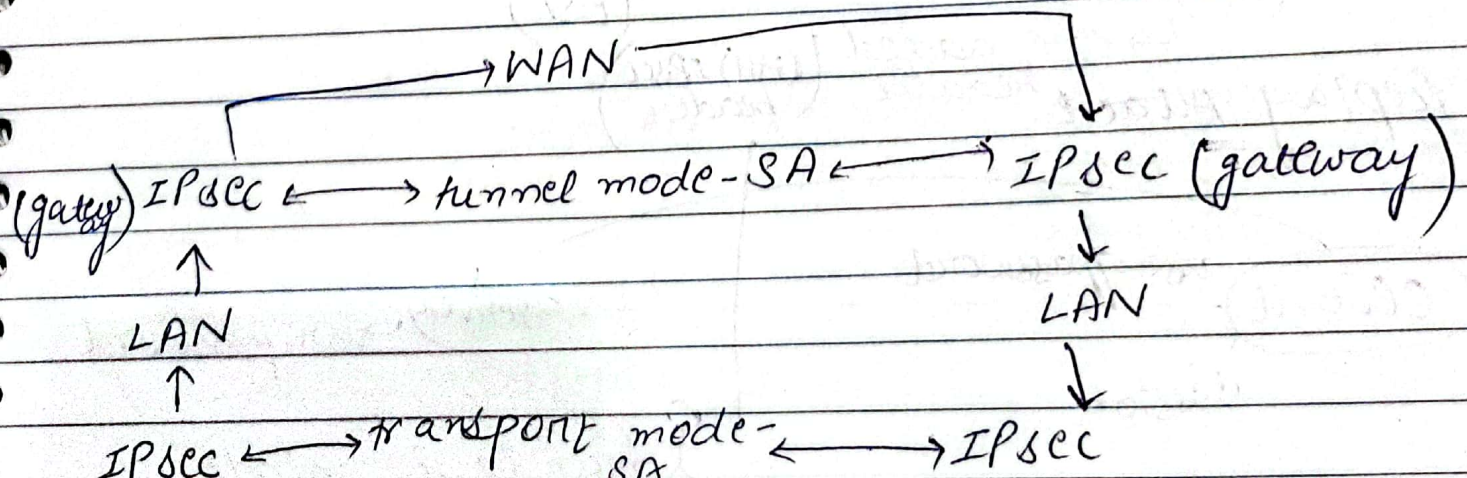
Lecture - 24

application → website → CSRF, XSS & tokens, SIP
transport → SSL/TLS/TCP ← +
network → IPsec
DLL
Physical

- IPsec ~ IP security → independent from application or higher protocols.
  - Security at network layer
    - within the operating system.

VPNs
↳ virtual private network

- switches (DLL + Physical)
  router (Network + DLL + Physical)
  endhost/server/client (All 5 layers)
- IPsec
  - transport-mode SA
  - tunnel-mode SA



•) Transport mode IPSec
↳ used for end-to-end security that is used by hosts.

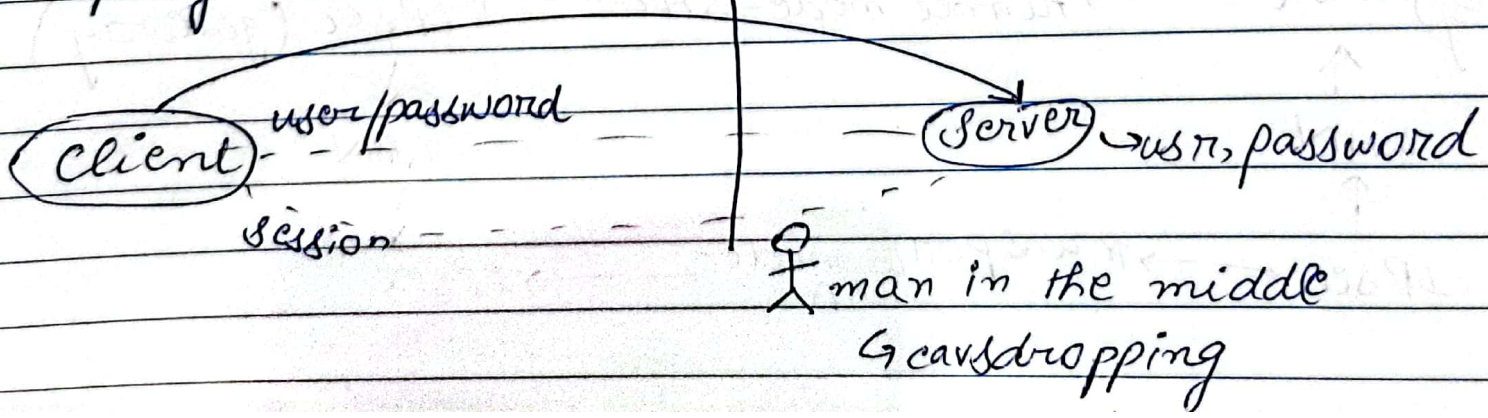
IPV4 header	IPSec header	TCP/UDP header + data

↳ adding in datagram

•) Tunnel mode IPSec
↳ gateway to gateway. used for creating VPN's.

•) Types of Packet
- Authentication Header (AH) ↳ one added header
- Encapsulating Security Payload (ESP) (AH) IPSec header
  - transport
  - tunnel ↳ used by VPN's

→ Replay Attack



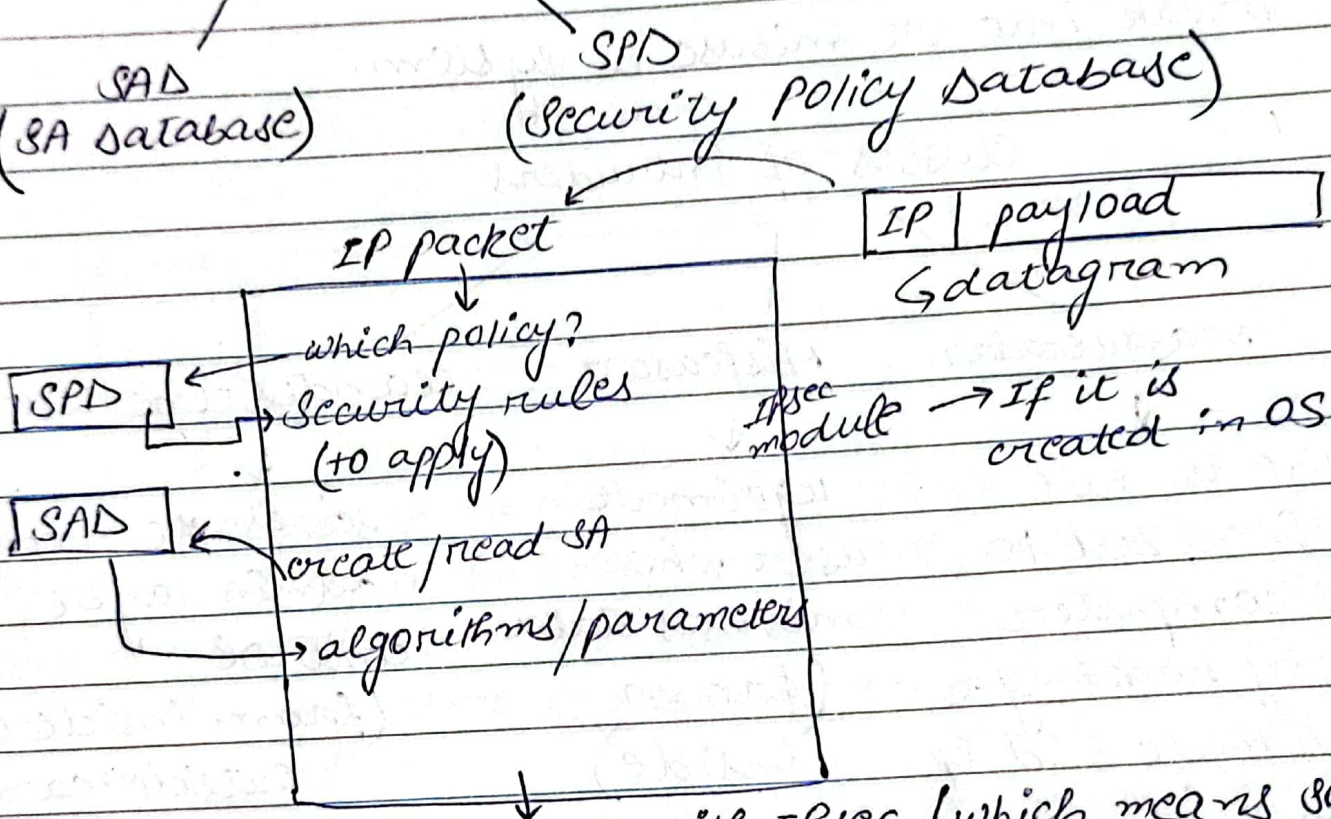
QoS → Quality of Service
↳ in network layer

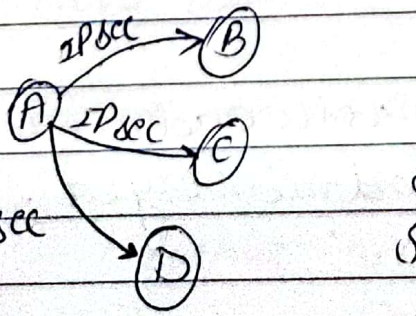
•) SA – Security Association
↳ establish shared security attributes b/w sender & receiver to support secure communication. It is uni-directional.

→ IP sec local databases

SAD (SA database)

SPD (Security Policy Database)





SA (A, B)
SA (A, C)
SA (A, D)

Foundation for Advancement of Science & Technology

Lecture - 25

- Intrusion detection system (IDSs)
  - Host-based
  - Network based

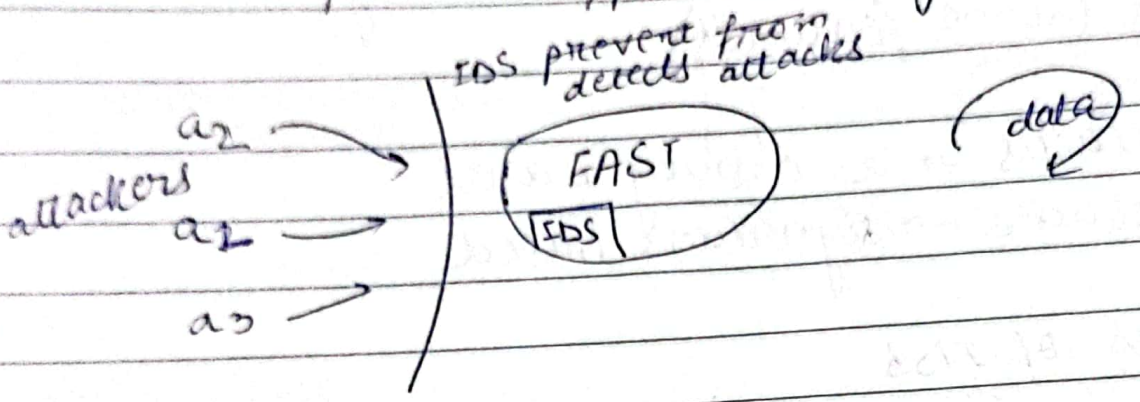
→ Intrusion
break into or misuse a system.

Classes of intruders

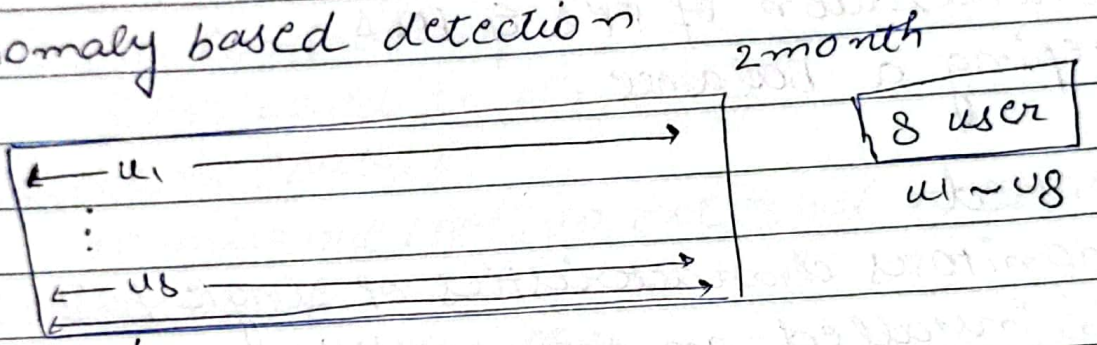
Masquerader	Misfeasor	clandestine user
↓	↓	↓
who is not authorized to use computer ~~as~~ by hacking a legitimate's id & using it (from outside)	legitimate user who misuses data (from inside)	seizes the root/ admin layer control (from inside or outside: can be both).

acc level compromise → username, password
root " " → root level, admin layer.

Intrusion detection system → passive
Intrusion prevention/protection system → active



→ Anomaly based detection



monitoring & noticing the network traffic & then using a strategy to observe / detect an anomaly based on their data. We get some usage pattern using some statistical analysis.

- IDS generate false alarms i.e False Positives
- Novel attack → (new type of attack)

→ Signature based detection
  ↳ set of rules on attack patterns
  ↳ what info is relevant
- based on simple pattern matching.

→ Hybrid based intrusion detection (→ mostly used nowadays)
↳ combo of anomaly based & signature based (some features)

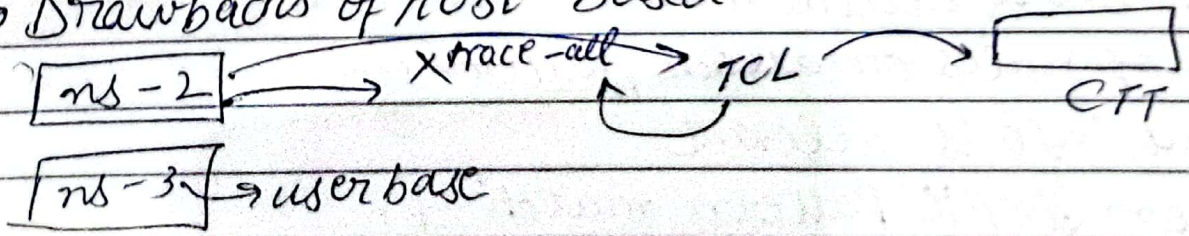
- noval attacks → anomaly based
- known attacks → signature based

Effectiveness of IDS
↳ minimization of FP's & FN's
↳ keeping a balance

→ Host-based
↳ monitors characteristics of single host
↳ IDS installed on end-hosts/server

→ Network-based
↳ attached to a router/firewall machine
↳ monitor network traffic for particular network devices/segments (application layer + transport layer + network layer - detection is done)

→ Drawbacks of HOST-based

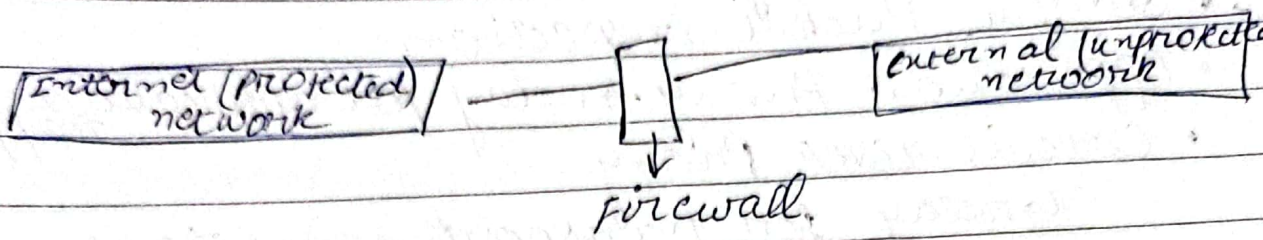


Foundation for Advancement of Science & Technology

Lecture-26

Firewalls

↳ used to protect networks.



→ Techniques

- Service control
  filter based on IP, port no
- Direction control
  to internal LAN, to external ⇆ Internet
- User control
  student vs faculty
- Behaviour control
  filter email with spam

→ Capabilities

- Single choke point
- location for monitoring security events
- convinient platform
- can serve as platform for VPN end-point (IPSec)

IXP → Internet Exchange Point

→ Types of Firewalls → accepts/rejects packets based on protocol header
- Packet filtering
- Stateful Packet Inspection
- Application Proxy → relay for application traffic
- Circuit-level Proxy
  - relay for transport connections

→ adds state information on what happened previously to packet filtering firewall.

Ex: ~~Packet Filtering + Stateful~~

Establishing TCP connection on circuit-level proxy.

Foundation for Advancement of Science & Technology

Lecture-27

~ Sandbox
↳ used for dynamic analysis

→ Access control
→ Confidentiality Model → BLP Model
(La Padula)
→ Integrity Model
(Biba, Clark Wilson)
→ Hybrid
(Chinese Wall)

→ Pen Testing (Penetration Testing)
↳ evaluating ~~all~~ the strengths of all security control.

- External vs Internal
- Overt vs Covert
  - ↓ Batakchi krna
  - ↓ Chupa kchi krna.

Lecture NO. 28"

BLP Model (Bell LaPadula) → multi-level security

↳ confidentiality → "no read up"
"no write down"

Biba Model

↳ integrity

→ "no read down"
"no write up"

→ Chinese Wall Model

↳ hybrid model

↳ (confidentiality & integrity) → authorized

→ conflict of interest (CI / CoI)

↳ a thing which causes problems that can lead to biasness

Elements

Subjects (active entities)

information

↳ objects (data items)

↳ datasets

↳ CI class

Access Rules.

→ BLP Model offers multi-level security because it follows properties like SS, * and DS (discretionary matrix)

→ Chinese Wall → History is maintained, access history

→ BLP → history-less (no history is maintained, no access to history)