Date
Page

Computer security:

→ Threat → potential for violation of security (eg: 'earthquake' is a threat that can exploit the vulnerability → 'cracked walls')
→ Vulnerability → weakness in system that can be exploited
→ Control (countermeasures) → technique to reduce threat, vulnerability, attack (prevent, detect, recover)
→ Risk → expectation of loss that a threat will exploit a vulnerability
→ Security policy → set of rules to protect sensitive & critical system resources
→ System resource (asset) → data, equipments
→ Adversary, attack (threat agent) (violate security policy)

* Security Principles: CIA
  - Two more properties
    - → Authentication (identity)
    - → Non-repudiation (accountability)
  - Computer assets → hardware, software, data, network

* Threat modelling:
→ identify assets, adversaries, motivations, threats, vulnerabilities, risk and possible defenses

Controls prevent threats from exercising vulnerabilities

Threat Consequences (3Ds):

→ Disclosure: → threat to confidentiality
  → exposure, interception, inference, intrusion
→ Deception: → threat to data or system integrity
  → falsification, repudiation, misuse, masquerade
→ Disruption: → threat to availability and system integrity
  → Incapacitation, obstruction, misappropriation, corruption

* Threat modelling eg: e-voting

Pol-election →	load ballot cryptographic file
Active voting →	use token to vote & votes are stored (encrypted) (single use)
Rural election →	stored vote transported to tabulation center

Confidentiality:
adversary shouldn't have access to voter's vote (who voted to who)
Integrity:
- no changing the vote
- no modification of ballot file
Availability:
- can't deny rights to vote

Social engineering:

Phishing : deceiving the victims to get sensitive info or install malware (critical money)

Date
Page

Security Goals (CIA):

viewing → modifying → using

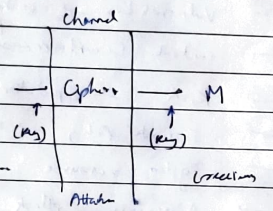
* **Confidentiality** : authorized access to info

→ Tools for confidentiality:

→ Physical security

→ Encryption

→ Access control (rules to limit access to confidential info)



Something a person
i) has (card)
ii) knows (pass)
iii) is (biometry)

→ Authentication (identity or role) (who the person is?)

→ Authorization : what the person has access to? (based on access control policy)

(eavesdropping) ↓ unauthorized interception of data transmitted

* **Integrity**: (alter in authorized way) (tamper)

→ Tools for integrity:

→ Backup,

→ Checksums (small change in input → different output)

→ data correcting codes (auto correction of errors)

* **Availability**: (info is accessible & modifiable in a timely fashion) (eg: timely response to our request)

→ Tools for availability:

→ physical protection

→ computational redundancies

* security implementation:

→ prevention, detection, response, recovery

* **Computer Security Strategy**:

→ security policy

→ security implementation

→ Evaluation

→ Assurance

Subject : person, process, program
object : data item
access mode : R/W/X
} Authorization : a policy

Date
Page

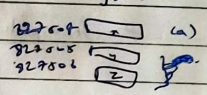
Program Security : → buffer overflow
→ TOCTTOU
→ Incomplete mediation

Flaws → fault (cause) (problem with a program)
→ failure (outcome)

$x = A/B$ (failure when $B = 0$)

* Software flaws :
- Out of bound read, out of bound write
→ Cross site scripting
→ SQL injection
- Improper input validation

char a = "xyz";



* Types of Security Flaws :

Intentional
- malicious
- Non-malicious

Non-intentional (always non-malicious) ← inadvertent
- Buffer overflow
- Incomplete mediation
- Time of check to time of use (TOCTOU)

Buffer Overflow :

```
char buffer[100]
gets(buffer);   or
strcpy(buffer, argv[1]);
```

| memcpy(void * dest, const void * src, size_t n)

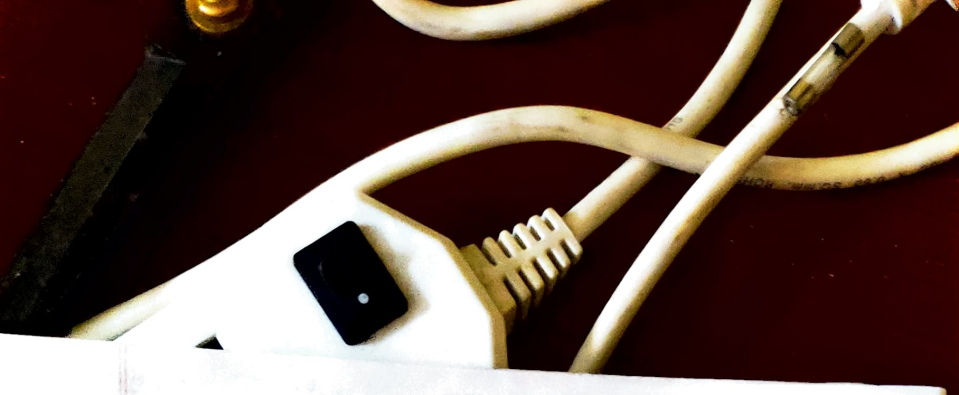
```
char sample[10];
sample[i] = 'A'
```

where i >= 10 → can change user data or OS data

* Consequences → program crash, run malicious code, corruption of program

```
int bof(char * str)
{
    char buffer[24];
    strcpy(buffer, str);
    return 1;
}
```

contains content of badfile (containing malicious code)



Date
Page

* Defense:
i) Compile time → canary (stack guard)
→ address space randomization
→ use language with bound checking
ii) Run time → non-executable stack

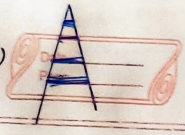
Incomplete mediation:
→ ensure user input is a meaningful request
→ client side mediation (eg form validation)
→ issue: user can modify client side stuff (eg edit form)
→ Incomplete mediation: sensitive data are in exposed & uncontrolled condition.
(eg. Police mentioned in URL)
→ Use server-side mediation

* Defence
→ input validation
→ don't let client return sensitive result
→ map / character list

TOCTTOU:
→ synchronization / serialization issue (race condition)
→ eg: logfile ← file owned by other (two permission for owner to write)
Between check & open, logfile ← etc/passwd
(symbolic link)
Now attacker can write entry into etc/passwd file

* Defense:
→ using digital signature & certificate
→ lock data values until checking (& before use)
→ all info related to access control decision is combined to TOC & time of access

Drive-by-downloads → downloads that happen (malware) without person's knowledge



Malware :

→ Virus, worms, backdoor, rootkit, trojan horses, spyware

* Execution of malware :

→ download & run malicious software
→ browsing infected pen-drive
→ opening executable email attachment
→ buffer overflow in email-clients or web browsers or network daemon

* Classification of Malware :

i) on basis of how it propogate
ii) payload : actions performed once it reaches target system

Virus :

→ self-replicating, needs host for spread
→ copy at beginning of executables
→ add itself as a macro in document (starts automatically when file opens)

* Virus residence :

i) bootstrap sector
ii) In-memory resident program
iii) in application programs (eg: startup macro executed when app starts)
iv) in libraries

→ Virus spread b/w computers : → sending infected files (pendrive)
→ put on p2p network

* Virus Phases :

– dormant, triggering, execution, propagation

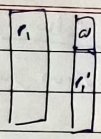
Date
Page

* Compressed virus:

search() → get files_to_infect (P_i)

infect() → Compress P_i, prepend CV (virus) to file

main() → (i) infect
(ii) decompress victim file & run



* Splitting Virus:

→ Signature based protection

→ behaviour based protection

* Polymorphic virus → modified copy in each target file

→ encrypting using random key

→ procedure decrypt is the signature visible to scanner

* Virus code with encryption:

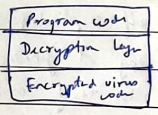
→ get files_to_infect P_i

→ Virus string ← entire virus code

→ encrypt Virus string → V'

→ prepend encrypted virus code, Key, decryption procedure to the file

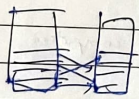
Target Program after infection:



* Metamorphic virus:

→ virus that can transform and change its internal structure rewriting & reprogramming itself each time it infects a new system

→ no encryption used



Worm:

→ self replicating, no host required, exploits security flaws to infect.

→ spreads through internet

2

Date / / 2022
Page 52

* How worms spread?

(i) remote shell facilities

→ sendmail : exploit debug option in sendmail to allow shell access

(ii) Cracking password:

→ Rsh : brute force cracking

(iii) Buffer overflow in networking software

→ Fingerd : buffer overflow in gets

* Morris worm :

→ strange log messages, strange files, large no. of process generated

Backdoor : Lecture 10 - 16 :

→ software that allows access to computer system bypassing normal authentication procedures. eg: sendmail worm & password hard coded in login program

* Rootkit :

→ malware that provides privileged, root level (administrative) access to a computer.

* Trojan

→ Malware that appears legitimate programs but actually are malicious. They claim to perform one function but actually do another malicious fn.

→ Methods of infection : → open infected email
→ click a link to malicious site
→ install by exploiting software vulnerabilities

* Spyware → Keyloggers
→ Screen-scrapers

} collects user info without his knowledge.

* Pegasus spyware :

→ access to phone's (android / ios) files, camera, location, mic, etc

* Fileless malware :

→ execute malicious JS / VB script directly in the memory

→ no file required, hard to detect, high complexity

Antivirus :

1st gen — simple scanners

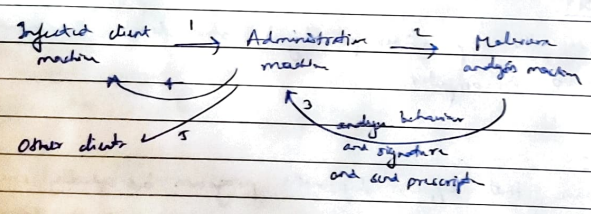
2nd gen → heuristic scanners

3rd gen → activity traps

4th gen → full featured protection

Digital Immune System :

→ comprehensive approach to virus protection



Malware Detection

Signature based

Anomaly based (behaviour)

Specification based (set of rules for what is a valid behavior)

→ static analysis — source code & binaries (file path)

→ dynamic analysis → during or after execution (monitoring behavior in sandbox)

→ file permission matrix

VA —translate table→ PA

VA = <segment num, object within segment>

segment table [segment num → base physical address & SIZE]

PA

offset ≤ SIZE for access

Issue → external fragmentation

Paging: VA = <page #, offset>

Page table (page # → frame #)

PA = <frame, offset>

Each address reference is checked for protection by hardware → Each page has bits for memory protection attributes

Issue → internal fragmentation

* Each process isolated from all others & from OS

→ Segmentation with paging in Linux

→ Memory protection bits indicate no access, RW access or read only

→ NX (non-executable bit): page instructions can't be executed

DAC:

owner of object can delegate permissions of the object to another user

ACL: File 1 → A (RWX, read, write) → B (read) → C (RX, no)

File 2 →

ACL (access control list)

→ Subject

→ Permission

→ Reference Monitor (checks permissions based on DAC policy)

ACL: collection sorted list of ACE (access control entry)

eg: ACL for a folder:

Bob; Read; allow

TA; Read; allow

Bob; write; deny

TA: write; allow

→ Access Control Matrix
→ Extended ACM

→ Password + salt
← h(), f(), r
f(r', h(p)) →

Date
Page

Verify

Linux File System :

→ Hard link
→ Soft link

* Linux vs Windows
allow only file ACL + execute another system
deny + allow only file ACL

* SUID : SGID, Sticky bit

* setfacl, getfacl

(self complimentary structure)

Date
Page

Symmetric Key Encryption → Block cipher → 64 bit / 128 bit
→ Stream cipher → bitwise (LF
→ bytewise (R

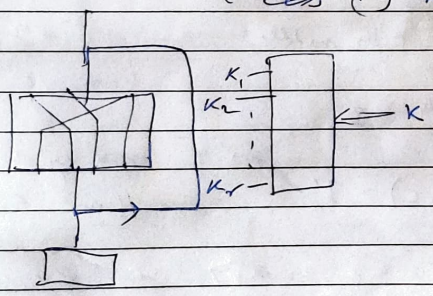
Key mixing
- Substitution
- permutation

Properties → Diffusion
→ Confusion

Diffusion → 50% bits changed in ciphertext by changing a bit in plaintext.

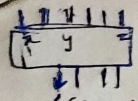
Confusion → plaintext ↔ ciphertext (non-linear & complex)

K —(key scheduling algo)→ we get r keys (say each of 128 bit)
$K_1, K_2 \dots K_r$
(Key K_i used in round i)



* Brute force attack → exhaustive key search
eg: key size = 128 bit ⇒ $\left(\frac{2^{128}}{2}\right)$ on avg operations
→ independent of algorithm of encryption

* Cryptanalytic attacks → exploits weakness of algorithm

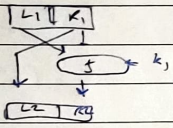


× 8 a

$\frac{64}{32} = \frac{6}{4}$

DES :

- 16 rounds
- 56 bit key K : Complexity = $O(2^{56})$ ← substitution square gives confusion
  $k_1, k_2, \ldots k_{16}$ → 48 bit (derived from K)
- 64 bit input / output
- Each round substitution + permutation
  $f(L_{i-1}, R_{i-1}, K_i)$



- For decryption use keys in reverse order

2 round DES : $K = (K_1, K_2)$ (56 bit, 56 bit) = 112 bit

$DES(K_2, DES(K_1, M))$ Complexity = $O(2^{56})$

Attack: Given → $<M, C>$ → find K_1, K_2

Attack: meet in the middle → $2^{56} + 2^{56} = 2^{57}$ (not 2^{112})

$DES_{enc}(K_1', M) = I$

$DES_{dec}(K_2', C) = I'$

→ If equal then found K_1 & K_2

Solution 3-DES : $<K_1, K_2, K_1>$ or $<K_1, K_2, K_3>$
but long computation time (3 times DES)

3 DES :

$DES_{enc}(K_1, DES_{dec}(K_2, DES_{enc}(K_1, M)))$ → 2^{112} $O(2^{112})$

K_3 → 2^{168} $O(2^{168})$

DES → 8 S-box (6 to 4)

DES → Feistal structure (reverse order of key work) (same structure)

AES:

→ Key size = 128 / 192 / 256

→ Input / Output → 128 bit

→ No. of rounds = 10 / 12 / 14

→ Round key size : 128 bit (fixed)

DES:
16+1/2 rounds
(only swap in last)
(no of (1))

→ optimized for encryption (non-Feistel)
(decryption takes a bit more time)
(inverse required)

Byte substitution ⟷ shift row (other double matrix)

Mix columns ⟷ add round key (⊕)

(these two can be used for encryption & decryption)

After 2 complete rounds → AES will have complete diffusion
(change one bit in input change 50% of bits in output)

Modes:

i) EBB → same key used for each input block
Problem : same ciphertext → same input

ii) CBC : (Cipher-block-chaining) :

$C_1 = E(K, IV \oplus P_1)$ → $P_1 = D(K, C_1) \oplus IV$

$C_2 = E(K, C_1 \oplus P_2)$

$C_n = E(K, C_{n-1} \oplus P_n)$

iii) OFB mode : Output feedback mode (Advantage → can be precomputed) — just XOR needed at the end

iv) CBC mode : Cipher feedback mode

v) CTR mode : can also be precomputed

LFSR → bitwise — | XTS mode for storage

RC4 → byte wise

$y = h(x)$ → secure channel

* MDC — manipulation detection code — x → unsecure channel

MAC — message authentication code — $h(x) = h(x')$ ✓

Properties of hash f°:

i) one way property / pre image resistance → $O(2^n)$

ii) 2nd pre-image resistance / weak collision resistant → $O(2^n)$

iii) collision resistant (strong " ") → $O(2^{n/2})$

MDS :

Message | Padding | L (length) — added (64 bits)

Max message length = 2^{64} bits

After padding, the messages do not have same hash

$H_i = f(H_{i-1}, x_i)$

100 | 000 / 101 | 0000 padding

H_0 = IV → random but known to everyone

padding → same hash

→ 128 bit digest (64 bit security)

→ each segment → 512 bits

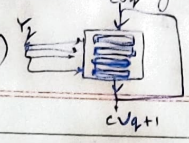
four rounds each of 16 steps

Your daily 57

f, g, h, i → boolean functions (one in each round)

$CV_0 = IV$

$CV_{q+1} = f(CV_q, Y_q)$ (boolean)



(64 steps / block)

SHA-1

→ 160 bit message digest (80 bit security)

→ each round : 20 steps

→ word size = 32 bits → IV → (A, B, C, D, E) (32 × 5 = 160)

→ 80 steps / block (slower than MD5)

→ Pad message such that its length ≡ 448 (mod 512)

→ optimized for big endian CPU

Message Authentication Code : MAC :

→ source authentication

→ integrity check (for both msg and hash)

K → not known to attacker

A → B (x || h_K(x)) Bob and Alice share a common key (known only to both of them)

$y = h_K(x)$

Attack : for (x, h_K(x)) find (x_i, h_K(x_i)) without knowing K

MDC → only integrity → no src. authentication

HMAC :

Why not $H(K \| M)$?

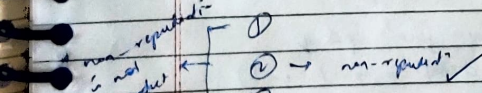
MAC → integrity + src authentication

(confidentiality is not our target)

Diffie Hellman :

(this problem) our approach has 1/6

Message authentication using a one-way hash f' :



② → non-repudiation ✓

integrity ✓
src authentication ✓
} all 3

MDC — Integrity

MAC — MDC + src authentication

Digital sig — MDC + ① + non-repudiation

Symmetric key cryptography:

Problems: i) Keys = $\frac{n(n-1)}{2} = O(n^2)$

ii) Key sharing

* Confidentiality + Integrity + Authentication →

A → B : Sign → Encrypt (M || σ) → better in terms of storage

(A's private key) (Bob's public key)

Otherwise: Encrypt → Sign (Computationally better to verify)

(signature verification is less costly)

RSA :

p, q

$n = pq$, $\phi(n) = (p-1)(q-1)$

Choose $e < n$: $\gcd(e, \phi(n)) = 1$

Find d: $ed \equiv 1 \pmod{\phi(n)}$

Extended euclidean algo

$d = e^{-1} \bmod \phi(n)$

$ed = 1 + k\phi(n)$

Enc → $C = M^e \bmod n$

Dec → $C^d \bmod n$

$= M^{ed} \bmod n$

$= M \cdot M^{k\phi(n)} \bmod n$

$= 1$ (Euler's theorem)

$M^{k\phi(n)} = 1^k \bmod n = 1$

$= M \bmod n = M$

Public key = (n, e)

Private key = (n, d) or (p, q, d)

If from n, we can find p & q

then we can find $\phi(n) = (p-1)(q-1)$

& hence I can find $d = e^{-1} \bmod \phi(n)$

Public key cryptosystems can be used for: (eg RSA)

→ Encryption
→ Key exchange
→ signature

Date
Page

Digital signatures:

Hash computation → trapdoor f^n

$H(M) = m$

$\sigma = m^d \bmod n$

$\sigma^e \bmod n = m^{ed} \bmod n = m^{k\phi(n)+1} \bmod n$

$= m \bmod n$

$= m$

Receiver calculates $H(M) = m'$ — Verifies if it's same

(Confidentiality not provided)

~~Soln~~ $(n, s) = (n, 42)$

$5^e \bmod n = 42^5 \bmod 91$

$= 42^2 \cdot 42^2 \cdot 42 \bmod 91$

$H(m) = 35$ $= 35 \cdot 35 \cdot 42 \bmod 91$

$= 42 \cdot 42 \bmod 91$

$= 35$

ElGamal Signature:

$x \rightarrow$ private

$\sigma =$ signature $= \langle r, s \rangle$

If same k (random) chosen twice, then problem

$r = g^k \bmod p$

$s = k^{-1} (m - xr) \bmod (p-1)$

Signature verification: $V_1 = g^m \bmod p$

$V_2 = y^r r^s \bmod p$

$V_2 = y^r r^s \bmod p$

$= g^{x r} g^{k s} \bmod p$

$= g^{xr + k(k^{-1}(m - xr))}$

$= g^m = V_1$

If $V_1 = V_2$ ✓

DLP: g^x (discrete log. problem)

Computationally infeasible to find x with given

$y = g^x \bmod p$

Date
Page

↪ Security based on DLP.

DSA (Signature algo):

p → very large (1024 bit)
q → small (384 bit)

public : p, q, g, y ($y \equiv g^x \bmod p$)

$v = ((g^{u_1} y^{u_2}) \bmod p) \bmod q$

$= g^{ms^{-1}} g^{xrs^{-1}}$

$= g^{(m+xr)s^{-1}}$

$= g^{(m+xr)\frac{k}{(m+xr)}}$

$= g^k$

$= r$

$H(M) = m$

$u_1 = mw \bmod q$

$u_2 = rw \bmod q$

$w = s^{-1} \bmod q$

$s = k^{-1} \alpha$

$s^{-1} = \alpha^{-1} k$

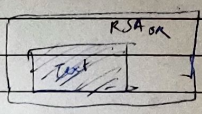
$= \frac{k}{\alpha}$

Symmetric key encryption → more efficient than public key encryption

Problem → sharing same key

Digital Envelope →

Rather than RSA for entire message, use AES (symmetric key)
(public key)



Text encrypted using AES (random key)
& the key encrypted using RSA (random)

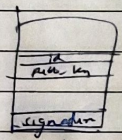
Public key directory :

Access control → we can add but not delete

Username	< public key >

Problem :
multiple same usernames in the directory

Solution. Public key certificates :



of certificate Authority

A → B

Alice || $Cert_A$ || σ

slide 13/24 → imp.

Date
Page

Lecture 11 :

* 4 basic types of threats

→ interception, modification, fabrication, interruption

→ Passive ~~Active~~ threats — Active ~~Passive~~ threats

Passive threats:
- release of msg contents
- traffic analysis
- (monitoring or eavesdropping info. without active engagement)
- less dangerous

Active threats:
- masquerade (impersonation to have unauthorized access)
- replay (sending some message again) (server assumes a new message)
- modification of msg contents
- DOS

* Insider Attacks (already have authorized access to target system) V/s Outsider attacks (attackers do not have authorized access to the system) (exploits vulnerabilities inside)

Threats:

→ reconnaissance : gathering info about target system / network to identify vulnerabilities, entry points for an attack.

→ Spoofing : impersonation or disguising identity by falsifying info to gain unauthorized access to a system
eg: i) modify src IP (IP spoofing)
ii) website spoofing : fake website appears as legitimate to gather sensitive info (phishing)
iii) DNS poisoning : modify the DNS cache to redirect a URL to a fake website (IP) rather than legitimate one.

→ attacks on confidentiality, integrity, availability

HAVELLS

is used to connect via wifi or ethernet
(~~is~~ embedded in mother board)
or can be external & programmed to intercept data

Date
Page

→ Attack on physical layer

→ Wiretapping : Intercepting / eavesdropping / monitoring electronic communication without knowledge or consent of party involved
(eg telephone, internet connection etc.)

Optical fibre : Harder than copper cables
no inductance.
↳ eavesdrop without physical contact

* Packet sniffers
* Port scanning → what systems are listening & reachable over the internet. eg tool → nmap, netcat

eg :) TCP port scanning:
TCP connect scan, TCP SYN scan, TCP FIN scan
ii) UDP port scan
↳ closed port : ICMP port unreachable.
Closed port : RST

Attacks on data link layer :

→ CAM table poisoning → modify table entries to redirect traffic to attacker.

CAM (Content Addressable memory) table or MAC (Media Access Control) address table maps MAC address to corresponding network ports.
↳ (of the switch)

(MAC address is stored in the NIC hardware) eg (00:1A:2B:3C:4D:5E)
(12×4 = 48 bits)

→ MAC flooding : overflow table to forward frames to all ports on a VLAN (like a broadcast)

Before :
Port 1 → MAC1
Port 2 → MAC2
Port 3 → MAC3

After
Port → MAC1, MAC2, MAC3

in switches
↳ multiport bridge
↳ filtering (ingress, classify, MAC, forward, broadcast)

ARP : used to find
Mapping from IP address to MAC address.

ARP table :

IP	MAC

Hub : incoming traffic msg is broadcasted to all nodes (no filtering)

Router
Switch
Switch
VLAN1
VLAN2
VLAN3

Virtual local area network (VLAN) :
logical division to isolate traffic
(broadcasting happens only with the VLAN)

Port 1 is actually only connected to Host 1.
If a msg is sent to Host 2 / 3 / 4, then it will be redirected to port 1 and ultimately to Host 1 (DOS)

End to end encryption.	Link to link encryption:
encrypted only at sender, decrypted only at destination	with transmission — encrypted, but decrypted and encrypted on each node.

* VLAN hopping attack:

— Unauthorized access to another VLAN.

Methods:

i) Switch spoofing — Attacking host imitates a trunking switch. (Happens when switch port defaulted to 'Dynamic Desirable')
→ Attacker can gain access to all VLANs allowed on the trunk port.
attacker's VLAN — victim's VLAN

ii) Double tagging : Two tags — VLAN 1 — VLAN 2

Attacker → switch 1 (strip off 1st tag) → switch 2 (sees the VLAN 2 tag) → forward to VLAN 2

* ARP poisoning attack: / ARP spoofing attack.

→ ARP : stateless protocol, no authentication of parties,
hosts caches all ARP replies send to them

→ Man in the middle attack (3rd party can read private data)
→ DOS (redirection of traffic)
→ MAC flooding (on switch) → switch bombarded with forged ARP packets (fake ARP packets) (fake version)
→ DOS by spurious packets
victim is busy processing the packets.

Someone asks who has IP x?
Attacker replies I have IP x my MAC is y.

Table entry:

IP	MAC
x	y

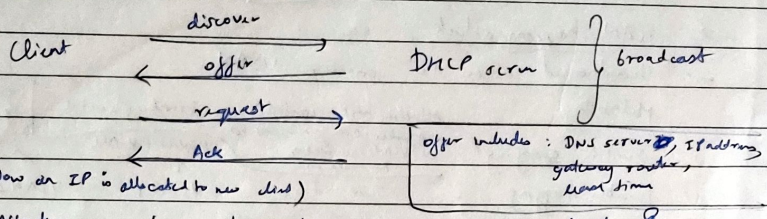
but actual MAC for x was y'.

* Mitigation & Detection:

→ Static ARP cache (table) entries
→ Changes in IP → mac mapping reported to admin.
→ Cryptographic techniques: secure ARP (authenticate), TARP → ticket based

and other network parameters

DHCP: (assign IP addresses to devices on the network.)



Attacker can hear broadcast message from a new client & race against the DNS server to give its own offer changing the actual DNS server / gateway router.

Threats

→ DHCP starvation:
DHCP server flooded with large no of DHCP request with spoofed MAC addresses. → DOS to legitimate devices

→ Substitute fake DNS server
→ Substituted fake gateway router
→ Man in the middle

Attacker can redirect traffic and eavesdrop on the network traffic.

IP Layer attacks (L3):

* IP spoofing:
→ Attacker changes the IP (source) (impersonates another user)
→ to gain root access to a victim host / backdoor entry
→ also called header forgery as header is forged with fake info.
→ Vulnerability: Any system using IP address as authentication method.

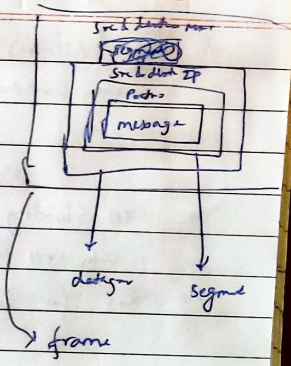
Other derivative attacks of IP spoofing:
→ Man in the middle → Smurf attack
→ Routing redirect → SYN flooding
→ Source routing → strict → DOS
→ loose

DOS → multiple requests to server using fake (spoofed) IPs. consumes resources preventing legitimate user requests

Date
Page

* Teardrop attack: DOS
Attacker sending a series of fragmented IP datagram pairs to the target to cause target system to crash.

attack when (end - offset < 0)



* ICMP Attacks:
↳ Internet Control Message Protocol
↳ protocol for network diagnostics & error reporting
→ to exchange control messages
→ no authentication

if frame size > MTU, fragmentation where each fragment be carried a separate frame with its own data link header & a fragment identification value for reassembly.

— ICMP 'redirect' message sent by attacker to the victim to instruct a new optimal gateway passing through the attacker & the attacker can sniff the communication

* Ping flood (ICMP flood):
Flood the victim with large amount of data packets
eg: 65500 bytes

* Ping of death:
Maximum size of ICMP echo packet = 65535 - 20 - 8 = 65507 bytes
If (Offset + size) > 64 KB → OS crash, reboot or hang
(fails to handle oversized fragments) (reassembly vulnerability)

* Smurf attack: DoS
Attacker spoofs ICMP echo request to a network broadcast address. (src IP = spoofed (of the victim)), dest IP = broadcast
All hosts will response to the spoofed IP address causing network congestion at victim.

TCP header

Lecture 15 : 2/24

Ping

Date
Page

source port + destination port + sequence no.

TCP Attacks:

3 way handshake

SYN →
← SYN+ACK
ACK →

Ping → to check if target host is reachable & responsive.
ICMP echo request →
← ICMP echo reply

* SYN flooding : (finite queue size for incomplete connection)
→ (Only SYN sent, ACK not sent)
→ DOS for legitimate users.

* TCP session hijack :
→ Hacker takes over a TCP session.
→ Attacker disconnects the host & inserts another machine with same IP address (TCP, UDP doesn't check validity of IP)
→ Man in the middle (watch sequence no. & ack. no.)
→ Blind Hijack → brute force all combinations of seq. no.
→ If attacker knows ports & sequence no., he can hijack and inject nasty data into the TCP connection.

Network Security Controls :

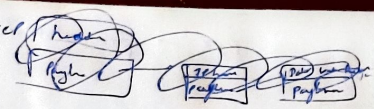
* Link layer → WEP, WPA, WPA2

* WEP: aim: confidentiality, data integrity, access control
actually : none enforced.

Key (IV || K) → 64 bit / 128 bit (24 bit)

IV ≡ V

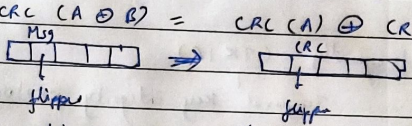
↓
pseudo random generator
↓
keystream (Z)



→ CRC : used for error checking & should not be used for integrity check.

→ Properties

→ $CRC(A \oplus B) = CRC(A) \oplus CRC(B)$

→ 

(If a bit of msg is flipped, corresponding bit of CRC is also flipped)

Key stream = $RC4(V, K)$

Encryption: $C = RC4(V,K) \oplus$ (message + ICV) — ICV: integrity check value (4 bytes)

Decryption: $\oplus$ $RC4(V,K)$ ~~EX Msg' CRC'~~

A → B : $(IV \parallel C)$

$= RC4(IV, K) \oplus [M \parallel CRC(M)]$

But Oscar sends $(IV \parallel C')$ to B which B will accept

where $(IV \parallel C') = RC4(IV,K) \oplus (M' \parallel CRC(M'))$

where $M' = M \oplus X$ (where X is some selected no.)

→ this will decrypt to M' & will be accepted by 'B' as CRC is still satisfied. But actually the data is altered ($M \to M'$) hence no integrity.

→ Also $C_1 \oplus C_2 = P_1 \oplus P_2$ (for same ~~IV and K~~)

So knowing one plaintext will get you other

+ Other problems : small key length (40 bits), lack of key distribution method, (RC4 with similar key → output keystream has weakness)

static encryption key (remains unchanged for long time) → if found, the attacker can decrypt the network traffic.

PSK → pre-shared key

WPA (Wireless Protected Access):

→ stronger data encryption & user authentication.

→ CRC-32 → MAC (message authentication code)

→ IV : 48 bit (WEP → 24)

→ Key changed frequently (TKIP)

Temporal Key Integrity Protocol

→ Key : 128 bit (WEP → 104)

WPA → WPA personal → PSK (weak)

→ WPA enterprise

WPA-2 :

→ replaces RC4 and MIC with CCMP algo which uses AES

→ counter based CBC-MAC protocol