LEEAD FINAL NOTECARD | OSU CS370 | FALL 2019 | SIDE 1/2

SECURITY NOTIONS: CONFIDENTIALITY: prevent unauthorized data access; Privacy: Conf WRT People; INTEGRITY: no writes to dat. AVAILABILITY: timely access to data; Accountability: trace to cause/source; **SECURITY MECHANISMS**: prevention, detection, response, recovery. **Terminology**: asset: something of value; threat: circumstances put sys in danger. snooping, spoofing for ex.; Adversary: materializes thre threat; Vuln: weakness; Attack: when adversary exploits Vuln; Att. Surf: all exploitable and reachable vulns. **Security Strategy**: 1] Policy/spec 2] implement 1 3] Correctness/Ass 2; Incentives (1-3). **Security Principles**: Economy of Mech: KISS; Fail Close: safe state; complete mediation: check everything; open design; Sep of Priv: 2 keys or PWS for critical ops; Least Priv; Least Common Mechanism; Psych. Adopt; Work Factor; Compromise Record.

CRYPTO: Encryption: confidentiality; privacy; Hashes: Integrity MAC: Integrity; Dig. Sig: INT, N-R, Authenticity; **Kerchoff/Shannon**: Assume Algorithm is public, keys are private; **Attack Types**: CT only: know CT, find PT/K; PT only: know PT and CT. find K; chosen PT, have enc. oracle, find K; chosen CT: decryption oracle, find K. **Cipher Types**: transpositional scramble; substitution: shift; product: both **Avalanche**: change 1 bit PT → great (>50) CT change); **Ciphers/Diag**:

stream $m = m_1 m_2$ K	Block $m = m_1 m_2 -$ $k = k_1 k_2 \cdots$
$E_K(m_n) = C_n$, same k	$E_K(m) = E_{K_1}(m_1)\, E_{K_2}(m_2) \cdots$

Encryption standards: P,K,C Length: DES: 64,56,64; AES 128, ~, 128. **DES and Dangers**: 3DES: 112 bit sec, despite 168 bits output, as MITM. Do E(D(E())) or spec'd method **Breaking DDES** where $C = E_{K_2}(E_{K_1}(P))$

P → [ENC] → 2^{56} Encod. | 2^{56} Decode ← [DEC] ← C
2^{56} keys ↗ (2^{56} keys ↙ DEC)
↳ this all has 2^{57} not 2^{112} — find match in decod / Encod

ECB: mode, use basic stream cipher, issue: identical pt → same C

CBC: IV m_0 m_1 — IV x m_0 → enc → c_0 x m_1 → enc → c_1 … | DEC: C_{n-1}, C_n → DEC x → m_n | Enc and Dec are with key K
↳ error only propagates for 2 blocks — self healing property

OFB: IV - ENC - K_1 - enc - K_2; m_1 x → c_1, m_2 x → c_2 | **COUNTER**: i — enc; m_n — x → c_i | **PubKey Principles**
- enc PK: confid.
- enc SK: int. Authen.
- Slower than Symm.

HASHING FXN: generate k bits from n bits; K≤n; Weak coll-r: find one match wrt; Strong CR: find two match; **Pigeon**: if inputspace > output, then collisions are guarenteed; all we need to try is ~~input~~ output size worth of inputs +1 to guarentee collision. **Birthday Paradox**: if hash has n bits, $2^{n/2}$ hashes → 50% collision **MAC**: Can add K to h(): $MAC_K(m) = h(K \| m \| K)$ **HMAC**: h() is hash, b → I (len), k has length b. ipad = 00110110, opad = 01011100 : HMAC(K,m) = h(K x opad ‖ h(K x ipad ‖ m)), **SHA512-HMAC**: HMAC(K,m) = SHA(K x ipad ∧ 8 ‖ SHA(opad ∧ 8 x K ‖ m)); Collisions dont matter for MAC: need to know K, and dont. **DIGSIG**: m ‖ E(h(m), SK_A); **PubKey Crypto**: RSA, elliptical can be used for all asym tasks and key distribution, signing and encryptions. Requirements: infeasable to break, easy to enc, dec, keygen. **Diffie Hellman KX**: based on discrete logarithm problem: n, g are int, P is prime, all 1024 bit ~. $n = g^K \bmod P$ →

everyone knows P, g. random K creates n. KX:
Person A: $PK_A = g^{KA} \bmod p$ ⟵SHARE⟶ $PK_B = g^{KB} \bmod p$ Person B
$K_{AB} = (P_{KB})^{KA} \bmod p$ — Same key if K_B K_A are flipped. vulnerable to MITM — attacker intercepts all traffic solved with PK certs, prob because no host ID

Totient Function: number of numbers less than n that are relatively prime to n (share no common factors) Ex: $\phi(10) = 4 \leftrightarrow \{1,3,7,9\}$. **Euler**: $x^{\phi(n)} = 1 \bmod n$; $n = pq$

RSA: choose 2 primes p, q, $n = pq$ ∴ $\phi(n) = (p-1)(q-1)$. choose $e < n$ s.t. e is rel. prim. to $\phi(n)$, computed s.t. $d \cdot e \bmod \phi(n) = 1$. $PK = (e, n)$ $SK = (p, q, d)$ encrypt: $m^e \bmod n = c$; decrypt: $c^d \bmod n = m$. **Hybrid Encryption**: send most symm, then use asym to send symm. key more efficient as $\varepsilon_{symm} > \varepsilon_{asym}$

AUTH AND FILE PERMS: **ACCESS RULES**: α = perm, * = transfer 1] if S_x with α*, S_x can transfer α[*] to *S 2] If S_x owns F, S_x may grant *α on F to *S 3] If S_x has control of S_y, S_x may remove *α from S_y 4] S_x can copy ACM for F they control 5] Any S can create F and grant *α to F 6] If S_x owns F, S_x may delete F 7] S_x may create S_y, then S_x owns S_y 8] If S_x owns S_y, S_x may del S_y. **Authentication**: bind external subject to internal (system) principal. **Complementation Information**: info you submit when you sign up; **Password Selection**: random phonemes, longer passwords and passphrases are Gud **Anderson's PW strength formula**: $P = (TG/N)$ P = prob suc T = time G = guess/unit time N = Charset size/num poss PWS. **Bloom Filters**: array of size of hash fxn output. (multiple h()) can be used. set bit in array if the h(x) is of x that is bad. can result in false pos but not wrong result. **Access Control**: Diagram, audit which

user ⊤ Auth f(x)/PEP → DB ↔ sysadmin; → resources
authentication ↳ authorization; **3A'S**: Authentication: binding x to internal; Authorization: grant access to resources; Auditing: independent reviews of sys act.'s **Access Control**: RB = role Based, D = discretionary, M = mandatory; AB = attribute Based. RB: based on role or group; AB: based on attributes and environment factors and context of access. DAC: uid ~ access rules, user can change; MAC: based on labels assoc with proc vs rules (TOPSECRET), user cannot change. **AC[L/M]**: ACM: by user, AKA capability, ACL: by object. **AC Requirements**: trust inputs, granularity of access, default closed, conflict resolution, admin policy, **AC Principles**: Least control, S.o.D, Dual Control: Change to AC* req 2x admin. **Linux**: setuid | setgid sticky (renames) | U, g, Other. **RBAC MODELS**: 0: pure, min. 1: 0 + hierarchies; 2: 0 + constraints; 3: 1+2. **RBAC 0**: (U) ═ [session] ═ (R) — Perms — (P) **RBAC 1**: Operates on (R) **RBAC 2**: operates every where

SIDE 2/2: **RBAC 1**: higher roles inherit below permissions. **RBAC 2**: can establish mutually x roles, cardinality of roles can be set. Prereq constraints support least priv. Static Sep D: SSD(RS, N) := no user assigned n or more roles from their rs. DSD := (rs, n)($n \geq 2$): no user session may activate $\geq n$ roles from rs. Role Engineering: B↑, U↓ Top down: use business rules to understand roles, B up looks at current AC, and uses ML to see roles. **BLP MAC**: confidentiality, users cannot be trusted. subj. are assigned a security level. **No Read Up**, **No Write Down**. DAC in MAC: also check ssp dac, both must be ok. * RBAC Advanced Sec Classes: add label, help least privilege. (A1, C1) d(A2. C2) iff A2 <= A1 C2 subs C1. S can read O if S dom O, S can write (app) O if O dom S. **Security Clearance Flexibility**: tranquility: immutable clearances and classifications. Weak tranquility (used): change wrt policy. **BIBA Integrity Model**: Integrity (as opp 2 confid.), higher integrity level, higher trust data is accurate/prog will execute as expected. Strict Integrity Property: $s \in S$ can write to $o \in O$ iff $i(o) \leq i(s)$ = **No write up**. Integrity Confinement property: $s \in S$ can read $o \in O$ iff $i(s) \leq i(o)$ = **No ~~write~~ read down**. Invocation Property: $s_1 \in S$ can invoke $s_2 \in S$ iff $i(s_2) \leq i(s_1)$. Integrity Labels: {$proj_1$, $proj_2$} = only for projects 1, 2. Integrity Level Comparison: (A_1, C_1) d(A_2 C_2) iff $A_2 \leq A_1$ and $C_2 \subseteq C_1$. SIP: S d O, ICP: O d S IP S_1 d S_2 **Chinese Wall**: conflict of Int. Rules: S can read o if {o in same DS as something already accessed by S ‖ o belongs to CI from which S has not accessed any data from} S can write to o if {S can read O ‖ all objects S can read are in the same DS as O}. **Multifactor Auth**: factors: (what you = #) #know #have #do (handwriting) #are where you are. **Biometrics**: stored in template, includes keystroke dynamics... **OTP S/KEY**: Seed K chosen by usr. Server calculates $h(K) = K_1$, $h(K_n) = K_{n+1}$ and saves $h(K_n) = (K_{n+1})$ ← saved. $P_1 = K_n$, $P_2 = K_{n-1}$, srv remembers last pw. Messages sent: ((s, d) info): (u, s) name | (s, u) {i} | (u, s) {P_i}. Server checks that $h(P_i) = h(K_{n-i+1}) = K_{n-i+2} = P_{i-1}$. If match, store P_i where P_{i-1} used to be. **HOTP**: Hmac OTP: shared SK "K", counter "C". HOTP(K, C) = Trunc(HMAC-SHA-1(K, C)) HOTP code = HOTP % 10^d | d = len. Counter++. **TOTP**: C_T = floor(($T - T_0$)/T_{step} = 30), HOTP(K, C_T). usually check ±1T. **Authentication Protocols**: PAP: unsecure: pass. Auth.; CHAP Challenge AP: handshake, also problematic. EAP: used, extensible; RADIUS, superceded by DIAMETER; KERBEROS: windows. **Nonces**: represented as R_n, = random integers, or time. unrepeated!! →

PAP DIAG: (U, S) {U, P} | (S, U) OK or NO. unencrypr.

CHAP DIAG: (S, U) "auth," chall | (U, S) username, h(chal, pw) | (S, U) OK or NO. Server accesses DB of pw's - danger.

NEEDHAM SCHROEDER: Establishes session key, using TTP = trusted 3rd party. TTP knows all keys. A, B; C = TTP: (A, C) A‖B‖R_1 | (C, A) {A‖B‖R_1‖K_S{A‖K_S}$_{KB}$}$_{KA}$ | (A, B) {A‖K_S}$_{KB}$ | (B, A) {R_2}$_{KS}$ | (A, B) {R2−1}$_{KS}$. In that, msg 3 can be replayed. fix: (A, B) "A" | (B, A) {A, R_3}$_{KS}$ | (A, C) A‖B‖R_1‖{A, R_3}$_{KS}$ | (C, A) {A‖B‖R_1‖K_S {A‖K_S‖R_3}$_{KB}$}$_{KA}$ | (A, B) {A‖K_S‖R_3}$_{KB}$ | (B, A) {R_2}$_{KS}$ | (A, B) {R_2−1}$_{KS}$.

SOFTWARE VULNS + OTHER: categories: input, output, OS interaction, code/logic. **3 BOF STEPS**: inj; CFH; Codex. **BOF Prot**: Memsafe lang's, safe coding, extensions or safe libs. stackguard, RAD, Guard pages, stack randomisation (ASLR), NX.

MALWARE: software that violates security policy. **Virus**: Attaches to programs, hops to others. can spread by drive by (most often similar to worms). **Trojan**: Something that is not what you expected/it said it is. **Logic/Time Bomb**: triggers on [logic/time] event. **Trapdoor**: allows unauthorized access to functionality. **Worm**: replicating virus that uses the network to propagate, and exploits a vulnerability in the target system. Dormant, spread, trigger, x-code. **Rabbit**: exhaust resources through replication. **Net Bot**: control-channel-orchestrated trapdoor. **Root Kit**: hooks OS calls to call Kit functions, stays hidden, changes syscall/function tables. **Virus Scanner Versions**: 1] signature-based 2] heuristics, integrity checks 3] behavior based 4] Multiple. **Zero Day**: no patch exists when vuln is disclosed. **BluePill Rootkit**: installed a hypervisor nested on top of existing one. **Virus Types**: Polymorphic: many same functioning copies of same virus; Stealth: hide from signature; Encrypted; Metamorphic: complete rewrite.

LAWERISM: **DMCA**: protect IP. **Laws about privacy**: 4th ammendment; Patriot Act; Wiretapping; Key escrow/PBS; FOIA; **Privacy Levels**: Anonimity: cannot determine identity; Pseudonimity: like Anon. can make user responsable for resource use; Unlinkability: can't tie 2 user acesses together. Unobservability: others can not see resource in use. **HIPAA** - health Sarbenes Oxley Gramm Leach Bliley - pressure on industry, - **GLBA** - finance

simple hash function: h(a) = (a+5) mod 7