Principle Of Easiest Penetration: Intruder Will Use Any Means Of Penetration..."Valuable Components": Hardware, Software, Data...EX: CARS: Tire Pressure Monitoring System Signal Went Wirelessly, Easily Eavesdropped With Unique Identifiers **Vulnerability:** Security Weakness That Might Be Exploited To Cause Undesired Consequences **Threat:** Asset Of Circumstances That Potentially Cause Loss Or Harm **Attack:** The Exploitation Of Vulnerabilities By Threats **Control: Protective Measure…** Threat Blocked By Control Of A Vulnerability **Types Of Threats:** Interception, Interruption, Modification, Fabrication **MOM: M**ethod, **O**pportunity, **M**otive (For Successful Attack, Attackers Must Have These)..Control? Eliminate One Of Them **Meaning Of Computer Security:** Should Provide: **C**onfidentiality **I**ntegrity **A**vailability **(CIA)...**Other Factors: Authentication, Authorization, Non-Repudiation, Privacy. **Hardware Vulnerabilities:** Easiest To Defend Against. EX: Add/Remove/Change Devices, Pull Plug, Reboot With Boot Disk To Use Machine For Attack, Mount HDs...Backhoe (Anchors)Vulnerability, Vulnerable Defibrillator With Wireless Access **Software Vulnerabilities:** Breaking/Delete Software, Modify To Do Something Different (Send Duplicate Transactions To Attacker) ,Software Theft… **Types Of Software Modification:** Logic Bomb, Trojan Horse, Virus, Trapdoor, Information Leak, Covert Channel.."**Vulnerability Life Cycle"**: Born (in Software/Hardware),Discovered, not Yet Patched (0-Day), Patched..MostVulnerableBeforePatch..0-Day:Black Market, Software Vendors Give Rewards **Data Vulnerability:** Understood By Lay People (SSN, Address, No skills Needed) ,Can Be Valuable (Private Company Info), Can Be Damaging If Modified (Air Traffic Control, Patient Drug Allergies) **Principle Of Adequate Protection:** Items Must Be Protected Only Until They Lose Value, Must Be Protected To Degree Consistent With Value (Data May Be Valuable Only For Short Time: Oscar Winner, Movie) **Data Compromised By:** Wiretraps, Bugs In Output/Input Devices, Monitoring Electromagnetic Radiation, Inferring One Data Point From Other, Just Asking...*80%Attacks Done By Insiders, Most Defense Mechanisms Designed For External Attacks **Adversaries:** Script Kiddies: Download Tools, Don't Understand Them. Amateurs: Average User Stumbles Across Vulnerability.. Cracker: Hack For Challenge. Career Criminals: Hack For Personal Profit. Users With Skills: Design, implement Tools **Method Of Defense:** Prevent: Close Vulnerability, Deter: Make Attack More Difficult, Deflect: Make Another Target Attractive, Detect: Know When Attack Occurs, Recover: Mitigate Attacks Effects. **Controls:** Encryption, Software, Hardware, Policies & Procedures, Physical **Principle Of Effectiveness:** Controls Must Be Used And Used Properly To Be Effective, Must Be: Efficient, Easy To Use, Appropriate **Principle Of Weakest Link:** Security No Stronger Than Weakest Link, Weakest Link Can Be: Firewall Power Supply, OS That Security App Runs Over, Human Who: Plans, Implements, Administers Controls **Confidentiality:** Only Sender, Intended Receiver" Understand" Message Contents **Message Integrity:** Sender, Receiver Want To Ensure Message Not Altered Without Detection **End-Point Authentication:** Sender, Receiver Want To Confirm Identity Of Each Other **Cryptosystem:** Cryptographic Algorithm, Set All Possible Plaintexts And Ciphertexts And Keys **Cryptographic Algorithm (aka Cipher):**Algorithms That Take A Key And Convert Plaintext To Ciphertext And Back **Mathematic Representation Crypto System:** K = {0, 1}^l ; P = {0, 1}^m ; C'= {0, 1}^n ,C ⊆ C' ;E : P × K → C ; D : C × K → P ; ∀p ∈ P, k ∈ K : D(E(p, k), k) = p ; It is infeasible to find F : P × C → K **Plaintext:** P(Characters, numbers, bits?) **Ciphertext:** C; **Encryption (encipher):**C = E(P) ; **Decryption (decipher):**P = D(C) = D(E(P)) **Cryptology**: Cryptography + Cryptanalysis **Cryptanalysis:** is the study of methods for obtaining the meaning of encrypted information without accessing the secret information("hacking")A good cryptosystem should be infeasible to: enumerate all possible keys, find the key from any reasonable amount of ciphertext and plaintext by enumerating possible keys, produce plaintext from ciphertext without the key, distinguish ciphertext from true random values **Kirchhoff's' Law:** "The system must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience." ; Secrecy must reside entirely with the key ; must assume that the enemy has complete details of the cryptographic algorithm ; enemy will reverse engineer your algorithm **Unconditional secure:** • If the ciphertext does not contain enough information to uniquely determine the plaintext • No matter how hard the opponent tries • One-time pad **Computational secure:** • If the cost of breaking the cipher exceeds the value of encrypted data • If the time needed to break the cipher exceeds the lifetime of data **Brute-Force Attack:** – Try every possible key on ciphertext until getting an intelligible translation into plaintext – On average, try half of the keys – Costly when key space is large – Remember Moore's Law **Secret key cryptography –** Involves the use one key **-** Bob and Alice share a same (symmetric) key – a.k.a. private encryption, single-key encryption, symmetric-key encryption ; or conventional encryption C = E(K, P) P = D(K, C) = D(K, E(K,P)) **Requirements for secret key cryptography** – Encryption algorithm is publicly known – Secure use of symmetric encryption implies: • a strong encryption algorithm • a secret key known only to sender/receiver – Need a secure channel to distribute keys! **Public key cryptography –** Involves the use of two (a pair of) keys **–** a.k.a. asymmetric encryption – Bob has a pair of public and private keys – Bob's public key is known by Alice • Alice uses Bob's public key to encrypt the message C = E(Kpub, P) P = D(Kpri, C) = D(Kpri, E(Kpub, P)) **Hash functions –** Involves the use of no key – Nothing secret – Hash algorithms are known as message digests or one-way transformations • Fixed-length, condense and one-wayness – Password hashing: secure password storage – Message integrity: keyed hash – Message fingerprint: digest – Digital signature efficiency **Caesar Cipher:** Every character is replaced with the character three slots to the right. Caesar cipher is a special case of shift cipher **Shift cipher –** Encryption: EK (m) = m + K mod 26 – Decryption: DK (c) = c – K mod 26 **Shift cipher is a special case of substitution cipher Substitution cipher** is to substitute one thing for another **Monoalphabetic cipher:** substitute one letter for another or keyword mixed alphabet **Polyalphabetic ciphers:** use multiple alphabets **Homophonic ciphers:** multiple possible output characters for an input character **Polygram ciphers:** encipher groups of letters at once Problem with monoalphabetic cipher? – One mapping scheme for the entire encryption process – Cryptanalysts could observe the patterns • Countermeasure – Use a different mapping for each character in the plaintext **The Vigenère Cipher** – Construct a table (the Vigenere tableau) – Each row in table is a different shift (alphabet) • Why shift cipher instead of monoalphabetic substitution? – Sender and receiver agree on sequence of rows – Helps to disguise patterns **Homophonic Ciphers** • Try to hide plaintext patterns (statistics) • Map each plaintext character to any of a set of ciphertext characters • Homophones : set of possible ciphertext characters that map to a single plaintext character. **Polygram ciphers** – substitute a group of characters for another group of characters – Goal: make it difficult for frequency analysis **Playfair Cipher** • Key table: all the letters into a 5 by 5 table – Treat I and J as one, or eliminate Q • Write the keyword (w/o duplicate)at the beginning • Encryption – Divide plaintext into pairs – Double characters separated by dummy character (x) – mi ss is si pp i becomes mi sx si sx si px pi – If plaintext has odd number of chars, append dummy char. • Encrypt plaintext pair-by-pair using the keybook. • **Substitution ciphers** – Monoalphabetic cipher – Polyalphabetic ciphers – Homophonic ciphers – Polygram ciphers – "classical ciphers" • Still vulnerable to various attacks – Brute force attack when key space is small • Vigenere Cipher (and other substitution ciphers) suffers from short keys – Brute force attack • **One-time pad** (Vernam Cipher) – Gilbert Vernam (AT&T): 1917 – Take a stream of random data (keystream) – Use it as the key to encrypt the plaintext – Message receiver uses same keystream to recover plaintext • If the stream is truly random -> perfect security! – Proven to be impossible to crack! • How to encrypt? – Bit-wise XOR – Shift (plaintext + key mod 26) **Rotor machines:** Implements a kind of Vigenere tableau – Keypad (to input plaintext) – Several rotors (to generate keys) – Keypad wired to a rotor and rotors wired to each other • Operation: – After each key is pressed, at least one rotor spins (gets a new mapping) – An output is generated for the pressed key – Rotors positions don't repeat until 26#rotors keys have been pressed • Effect: 26#rotors substitution alphabets • Sender and receiver agree on an initial state of the rotors (key). • Most rotor machines are designed to be involution: encrypt the ciphertext with the same initial settings generates the plaintext. • Most famous rotor machine: Enigma **Transposition Ciphers:** Rearrange the plaintext to get ciphertext**; Columnar Transposition:** Use a two-dimensional array (matrix)– Write plaintext in rows – Read ciphertext in columns **Combinations of Approaches:** Use a combination of the two → product cipher – Substitution adds confusion – Transposition adds diffusion **– Shannon Secrecy** P (M = m | E(K, m) = c) = P (M = m) • Probability of guessing the plaintext knowing the ciphertext = probability of guessing plaintext without knowing ciphertext. P (E(K, m) = c) = P (E(K, m') = c) • Probability of any message giving a ciphertext is the same • **Confusion and Diffusion** – Confusion: make the relationship between the plaintext and the ciphertext (or the ciphertext and the key) as complex as possible. • Use the key in a very complex way. – Diffusion: dissipate the statistical structure of the plaintext in the long range statistics of the ciphertext. • Have many plaintext characters (bits) affect each ciphertext character (bit) **Claude Shannon introduced idea of substitution-permutation (S-P) networks:** – The basis of modern block ciphers – S-P networks are based on the two primitive cryptographic operations: • substitution (S-box) • permutation (P-box) • we will see them in DES (1974) – Provide confusion and diffusion of message **• Stream ciphers** – encrypt one symbol (bit, letter) at a time – encrypt the i th symbol with the i th part of the keystream **• Block ciphers –** Encrypt larger blocks of plaintext – Encrypt all blocks with the same key – E.g. the transposition cipher example: • Encrypt 4 letters at once • Cannot just encrypt letter 1 – need to wait for the other letters in the block **Stream ciphers** – Vernam (one time pad) – Vigenere with period p – Rotor machine with r rotors: period of 26r **• Block ciphers –** Transposition ciphers with period p – Playfair **Stream ciphers –** Advantages: fast; low error propagation – Disadvantages: low diffusion; vulnerable to insertions and modifications **• Block ciphers –** Advantages: high diffusion; more immunity to insertion – Disadvantages: slower; error propagation **DES(Data Encryption Standard)::** – Block cipher. 64-bit blocks – same algorithm used for encryption and decryption – 56-bit keys (effective key length: 56!!) • represented as 64-bit • but every 8th bit is for parity only – symmetric: receiver uses same key to decrypt; Encryption Provides – confusion (substitutions) – diffusion (permutations) Same process 16 times per block **Strength:** Not that strong anymore, 1998<3days, NSA can now in milliseconds **Each Round:** Li = Ri - 1 ;; Ri = Li−1⊕ f (Ri−1 , ki) **Fiestel(F)Function**: Purmutauion, S-Box, P-Box **S-boxes (substitution boxes)** – R: from 48 bits to 32 bits – Break R into 8 blocks – 6 bits/block – S-boxes are different **P-box** – Input: 32 bits – Bits are rearranged according to a fixed permutation. – Output: 32 bits – Add diffusion – Each S-box's output bits are spread across 6 different S-boxes in the next round **Exclusive Or is Key Schedule:** Key is 56 bits (64 – 8 parity) **AES(Advanced Encryption Standard):: Block size:** 128 bits=16bytes **In each round** – SubBytes: non-linear byte substitution – ShiftRows: circular byte shift in each row – Mix Columns: add diffusion – Add Round Key **"State"** of machine given by 4x4 array of bytes. **SubBytes:** Change each byte of state with corresponding byte from SBOX matrix: SBOX [X,Y] Non-linear, based on polynomial arithmetic **Shift Rows:** • 1 st row is unchanged • 2 nd row does 1 byte circular shift to left • 3rd row does 2 byte circular shift to left • 4th row does 3 byte circular shift to left **Mix Columns:**• each column is processed separately • each byte is replaced by a value dependent on all 4 bytes in the column **Add Round Key:** • XOR state with 128-bits of the round key • again processed by column (though effectively a series of byte operations) **Strength:** NIST estimated that a machine that could break a 56-bit DES key in 1 second would take 149 trillion years to crack a 128-bit AES key **Diffie-Hellman Key Agreement:** – Started the modern age of cryptography – Enable negotiation of a secret over an insecure media – Idea: participants exchange intractable puzzles that can be solved easily with additional information. **Protocol:** *Seutp: both agree on a large prime p and a generator g (belongs To)Z_p ; Both are public info. EX. P=13, g=4 *Step1: Each principal picks a private value xa.xb(<p-1), and generates a new value ya= g^{xa}mod p, and yb = g^{xb} mod p *Step2: Exchange y, generate the secret shared key za = yb^{xa} mod p, and zb= ya^{xb} mod p *Step3: the agreed session key is g^{xaxb} mod p **Attacks on D-H:** This is a key agreement, not authentication – You don't know anything about who you have exchanged keys with – Insecure against active attacks, e.g., man-inthe-middle • Alice and Bob think they are talking directly to each other • Mallory is actually performing two separate exchanges **RSA(Rivest, Shamir, Adleman): 2002 Turing Award:: RSA Key generation** – Select two large primes p and q; (p != q) – Calculate n = pq – Calculate φ(n) = (p-1)(q-1) • Euler's totient function. – Select a random integer e, 1<e<φ(n),and e is relatively prime to φ(n): gcd(e, φ(n))=1 – Compute d, 1<d<φ(n), and d ≡ e -1 mod φ(n) • de≡ 1 mod φ(n) **Public key: <e,n> Private Key: <d,n> Note:** p, q, and φ(n) should be thrown away • **RSA Encryption** – Given: message m, 0<m<n, public key <e,n> – Compute c = m^e mod n • **RSA Decryption** – Given: ciphertext c, and private key <d,n> – Compute m = c^d mod n – Actually: c^d mod n = m mod n **RSA is thought to be secure because**: – to find d (inverse of e mod φ(n)) • need to know φ(n) – given n it's very difficult to find φ(n) • thought to be no easier than factoring n • Quantum computers and Shor's algorithm? • Note: when p and q are 100 decimal digits – n is about 200 decimal digits – millions of years of computer time needed to factor **Hybrid scheme (public + session key)** – Public key crypto is slow – Symmetric key is fast but key distribution problem – solution: • Create a symmetric key called session key • Encrypt the data with the session key • Encrypt the session key with the receiver's public key **MAC(Message Authentication Codes)::** Small block appended to message for authentication MAC = CK (M) – C mac function – K shared secret key – M message • The block is called – cryptographic checksum or – Message Authentication Code (MAC) • MACs verify – that the message came from A – that message has not been altered **Hash functions** take a message as input – Message may be of any length – Output is string/number of fixed length • Sometimes called message digest functions • Hash result: called digest or fingerprint • Cryptographic hashes ≠ function used in hash tables • Cryptographic hashes are one way: – Given M, it's easy to compute H(M) – Given H(M), should be very difficult to produce M – or any M' where H(M') = H(M) • "Collision" – Implies uniform distribution of hash values • Example cryptographic hashes: – MD5 – 128 bits – SHA1 – 160 bits **Hash Functions for Authentication** – Can we use a hash function as an authenticator? • Just send M + H(M) • No: Bad guy will send

M' + H(M') • Again: the hash function (crypto algorithm) is public – Try this: • Send: M + EA_pri(H(M)) • Or: H(secret + M) • Can I use Bob's public key? • Send: M + EB_pub(H(M)) **Authentication::** *We consider two authentication scenarios – Server authentication • Certificate – User authentication • With OS • In a distributed system **Public Key Infrastructure (PKI)**Goal of authentication: bind identity to key • Public key: bind identity to public key – Crucial as people will use key to communicate with principal whose identity is bound to key – Erroneous binding means no secrecy between principals – Assume principal identified by an acceptable name • Certificate: token (message) containing – Identity of principal (here, Alice) – Corresponding public key – Timestamp (when issued) – Other information (perhaps identity of signer) – Compute hash (message digest) of token • Hash encrypted by trusted authority (here, Cathy) using private key: called a "signature" • Bob gets Alice's certificate – If he knows Cathy's public key, he can validate the certificate • Decrypt the encrypted hash using Cathy's public key • Re-compute hash from certificate and compare • Check validity • Is the principal Alice? – Now Bob has Alice's public key • Alice is endorsed by Cathy! • Problem: Bob needs Cathy's public key to validate certificate – That is, secure distribution of public keys – Solution: Public Key Infrastructure (PKI) using trust anchors called Certificate Authorities (CAs) that issue certificates • **Hierarchical CAs with cross-certification** – Multiple root CAs that are cross-certified • Web Model – Browsers or Operating Systems come preconfigured with multiple trust anchor certificates – New certificates can be added (be careful!) – Bad certificate can be revoked. • Distributed model (e.g., PGP) – No CA; instead, users certify each other to build a "web of trust **Passwords:** What if the bad guy gets your password file? – Aside: this has happened a lot. Many of them through SQL injection attack. • Instead of storing cleartext passwords – Store passwords transformed through some one-way function, e.g. the hash of the password. • When user sends password – take hash of the password, H(pass) – check H(pass) == what's in password file Password crackers – Brute force – Dictionary based • What if the bad guy manages to get the a hashed password h(p) – Hash the terms in the dictionary, and compare them with h(p) – Counter measure: we can make the h() function very slow, or hash it many times. • If it takes 0.1 second compute hash – doesn't matter in real world applications. • However, the adversary could only test 600 passwords in a minute. **Rainbow table**: pre-computes h(p) for all entries in the dictionary **Counter measure: salt** – Add randomness to password hash – Random data called salt **Distributed Authentication**: In a distributed environment? – Key management – Secret key: if there are N principals, N2 shared keys would be needed – PKI: if there are N principals, N (public, private) key pairs would be needed **Kerberos:** A centralized authentication service – Authenticate users to services – Authenticate services to users – Servers are relieved of the burden of maintaining authentication information **History** – Developed as a part of Project Athena at MIT – Solves: password eavesdropping – Online authentication: variant of Needham-Schroeder – Easy application integration API – First single sign-on system – Sign-on once, access all services **• Most widely used (non-web) centralized password system in existence –** Adopted by Windows 2000 and all later versions – Also available for Unix/Linux family of OS – Again, there are some interesting stories about military adoption and export control… **The design of Kerberos –** Goal: reduce the amount of information each individual system needs to maintain. – Trusted third-party issues certificate – Prove that I am the person on the certificate – Certificates contain other properties useful for authorization decisions – Assumption: the communication channel is insecure. **The Kerberos architecture –** KDC: Key Distribution Center • KA : Shared between Alice and KDC *KDC: only Alice could see K; only Bob could see {Alice, K}**.** • KDC: only Alice could see the session key; only Bob could see the ticket. • Alice: only Bob could "understand" the ticket; only Bob could see m • Bob: only Alice (with session key) could see m+1 **Design rationales behind Kerberos –** Less work on servers (KDC and Bob), more work on clients • A client requests credentials (ticket + session key) and manage them; KDC and Bob do not maintain states about user credentials (except that it must remember authenticators seen in a time window to avoid replay attacks) • More scalable in a large distributed system – Less communication overhead • Alice sends both the ticket and the authenticator to Bob. Neither Alice nor Bob needs to wait. **Short-term credentials** – Kerberos tickets and session keys are short-term credentials. Unlike the long-term credentials (keys shared with KDC), they can be stored in memory or disk. **One ticket for one server** – The ticket is only good between Alice and Bob – Alice would have to type in password to obtain a new ticket for a different server • **Single Sign-On (SSO)** – Ticket-granting Server (TGS) • A dedicated server that issues tickets for all other servers – User only needs to type in password once to obtain a credential (ticket+session key) for TGS (Ticket-Granting Ticket) – Subsequent ticket-granting requests will be conducted through TGS **Kerberos Single Sign-On (SSO)** – AS: Authentication server – TGS Session Key: $K_{S\text{-}TGS}$ – Sending $K_{S\text{-}TGS}$ to Alice: $\{K_{S\text{-}TGS}\}_{KA}$ – Ticket Granting Ticket (TGT): $\{Alice, K_{S\text{-}TGS}\}_{KTGS}$ **EECS u**ses SSO Kerberos sytem **;; Database Security:: CIA:** Confidentiality: Prevent/detect/deter improper Disclosure of information ; Integrity: Prevent/detect/deter Improper modification of information ; Availability: Prevent/detect/deter improper Denial of access to services. **Access Control:** • Ensures that all direct accesses to object are authorized • Protects against accidental and malicious threats by regulating the read, write and execution of data and programs • Requires: – Proper user identification – Information specifying the access rights is protected form modification **Access control components: – Access control policy** specifies the authorized accesses of a system **– Access control mechanism i**mplements and enforces the policy**; Mandatory Access Control (MAC):** • Utilize security classifications – TS: Top Secret, S: Secret, C: Classified, U: Unclassified – TS > S > C > U • Each subject and object are classified into one of the security classifications (TS, S, etc.) • Bell-LaPadulla properties (restrictions on data access) – simple property: No READ UP – star (*) property: No WRITE DOWN (write at own level) **Multilevel relational (MLS) schema: (**Homework assignment with classifications) What if the S level wants to update one of the tuples at the U level? – U cannot see the update – Replicate the tuple **Discretionary Access Control (DAC):** • For each subject access right to the objects are defined – (subject, object, +/- access mode) – E.g. (Black, Employee-relation, read) • Based on granting and revoking privileges • Assign privileges – to account level (subject) • independent of the relations • create schema, create table, create view – on relation level (object) • on a particular base relation or view **Authorization ID's:** • A user is referred to by authorization ID, typically their login name. • There is an authorization ID PUBLIC. – Granting a privilege to PUBLIC makes it available to any authorization ID. • The objects on which privileges exist include stored tables and views. • Other privileges are the right to create objects of a type, e.g., triggers. **Privileges:** • A file system identifies certain privileges on the objects (files) it manages. – Typically read, write, execute. • SQL identifies a more detailed set of privileges on objects (relations) than the typical file system. • Nine privileges in all, some of which can be restricted to one column of one relation. **• Some important privileges on a relation:** 1. SELECT = right to query the relation. May apply to only one attribute. 2. INSERT = right to insert tuples. 3. DELETE = right to delete tuples. 4. UPDATE = right to update tuples. May apply to only one attribute. **Granting Privileges:** • You have all possible privileges on the objects, such as relations, that you create. • You may grant privileges to other users (authorization ID's), including PUBLIC. • You may also grant privileges WITH GRANT OPTION, which lets the grantee also grant this privilege. **• To grant privileges, say:** GRANT <list of privilages> "\n" ON <realtion or other object> "\n" TO <list of authorized IDs> • If you want the recipient(s) to be able to pass the privilege(s) to others add: WITH GRANT OPTION **Grant:•** Suppose you are the owner of Sells. You may say: GRANT SELECT, UPDATE(price) ON Sells TO sally; • Now Sally has the right to issue any query on Sells and can update the price component only**. Grant option** • Suppose we also grant: GRANT UPDATE ON Sells TO sally WITH GRANT OPTION; • Now, Sally not only can update any attribute of Sells, but can grant to others the privilege UPDATE ON Sells. – Also, she can grant more specific privileges like UPDATE(price)ON Sells. **Revoking Privileges**: REVOKE <list of privilages> "\n" ON <realtion or other object> "\n" FROM <list of authorized IDs> • Your grant of these privileges can no longer be used by these users to justify their use of the privilege. – But they may still have the privilege because they obtained it independently from elsewhere. • We must append to the REVOKE statement either: 1. CASCADE. Now, any grants made by a revokee are also not in force, no matter how far the privilege was passed. 2. RESTRICT. If the privilege has been passed to others, the REVOKE fails as a warning that something else must be done to "chase the privilege down." **RBAC(Role-based access control)**: – Semantic construct – System administrator creates roles according to job functions • Role – Specific task competency – duty assignments – Embody authority and responsibility • Grant permissions to users in these roles – Roles & permissions – Users & roles • Roles define individuals and extent of resource access • Combination of users and permissions can change – E.g. user membership in roles • Permissions associated with roles stable • Administration of roles rather than permissions • Role permission predefined – Easier to add/remove users membership than create new roles/permissions • Roles part of SQL3 • Supported by many software products – Roles used in Windows NT, XP (system admin) • Access control in RBAC exists in: – Role-permission (stable) – User-role (dynamic) – Role-role relationships (stable) • RBAC supports principles: – Least privilege – Separation of duties- mutually exclusive roles – Data abstraction- abstract permissions (not just R/W) • Limitations – RBAC cannot enforce way principles applied – system admin could configure to violate • Mutually exclusive roles – User at most 1 role in ME set – Combinations of roles and permissions can be prohibited • Cardinality – Maximum number of members in a role – Minimum cardinality difficult to implement • Prerequisite role – User assigned to role B, only if assigned to A – Permission p assigned to role only if role has permission q **DAC, MAC vs. RBAC**: • DAC vs. MAC emerged from defense security research • RBAC independent of access control • RBAC can be used to implement DAC, MAC **Database Encryption**:: **Application Level Encryption**: • Data protected in database & storage • Data protected in use and transit • Programming needed in the applications • Power of database limited -indexing, searching, stored procedures • All access must go through application • Key management **Database Encryption**: • Protects data as it is written to and read from a database • Secures data in the file-system used by database • Enables field level encryption • Transparent to applications • Watch data in transit **The Inference Problem**: • Inference problem is to infer or derive sensitive data from non-sensitive data. **Tracker Attacks**: • divide queries into parts – C = C1 AND C2 – count(C) = count(C1) - count (C1 AND ~C2) • combination is called a tracker • each part acceptable query size • overlap is desired result • Tracker attacks: Generate the desired data by using additional queries that generate small results. SELECT COUNT(*) FROM Sample WHERE Sex = 'F' AND Race = 'C' AND Dorm = 'Holmes'; • Use rules of logic and algebra to rewrite query: count (a AND b AND c) = count(a) - count(a AND NOT (b AND c)) = count(a) - count(a AND (NOT b OR NOT c)) **Controls for Inference Attacks**: • Query controls – Limit overlap between new and previous queries • Partitioning – Cluster records into exclusive groups and only allow queries on entire groups • Item controls – Suppression: query is rejected without sensitive data provided. – Concealing: the answer provided is close to but not exactly the actual answer. • No perfect solution • Three paths to follow: – Suppress obvious sensitive information (easily). – Track what the user knows (costly). • They are used to limit queries accepted and data provided. – Disguise the data (problem with precision). • It's applied only to the released data.