|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cryptanalytic attack分析明文、密文或key  Brute-Force暴力一波  DES key too short(56)→暴力  3DES key變長但data一樣64、加密太慢→沒有效率  1.ECB很多block用同一個key  2.CBC ECB+前後block chain(XOR)起來  3.Stream cipher:一個byte流進來和random key加密xor  →Faster/ far less code /適合用在網路  **Message authenticate**內容不要改，來源authentic  **Q：Symmetric key NOT suitable for data authentication?**  No need to be reversible / waste of processor resource  廣播給所有人，都要解密太麻煩了→**產生tag**  1. MAC(Message authenticate code)加密  2. One-way Hash不用key  3. Hash+encrypt先hash在symmetric/public encrypt  4. Hash **w/o encrypt** 對稱的Key夾msg一起hash  N是hash完的長度  A. one way(preimage)H(x)=h 2^n  B. Second preimage(weak collision)H(y)=H(x)given x 2^n  C. Collision(strong)H(x)=H(y) 2^(n/2)  **SHA(secure hash alg)** authenticate/digital signature/pswd  Public key：easy compute/public key不能猜到msg private key  **Digital signature:** hash and encrypt(可以CA自己做)  public key加密(its certificate=key+ID)  **Symmetric key exchange**(Hellman)需要authentication  **Digital envelopes:**  Msg用random symmetric key(用public key加密)加密  ------------------------------------------------------------------------------  **E-Authenticate**  Registration authority註冊 claimant Relying party登入(已註冊)  Credential service provider憑證 Verifier驗證  Credential護照 Token身分證ID  Offline dictionary attack離線字典攻擊  1. 離線拿到系統hash密碼檔  2. 破解hash value  3. 得到常見passwords  Specific account attack特定帳號開猜密碼  Popular pswd attack試常見的密碼  Pswd quessing against 1 user了解他開猜  Workstation hijack工作站不自動登出，滿危險的  Exploiting user mistakes很智障給密碼  Password sniffing攔截一波  用salt+pswd 一起hash很難  Q：For a salt of length 𝑏bits, the number of possible  passwords is increased by a factor of 2^𝑏  UNIX用DES x25/ MD5有salt+inner loop  Reactive pswd checking週期性跑cracker發現密碼不好就跟你說  Proactive註冊的時候確認密馬好不好(Bloom filter)   |  |  | | --- | --- | | Memory card(ex 磁條) | Smart token | | Need special reader  Token loss  User太多不夠用  (缺點) | 有Authentication protocol  Static  Dynamic  Challeng-response |   **Biometric Auth**  Verify他知道你是waterso就去找你在的地方，看指紋OK不OK/ Identify(不知道你是誰，掃database)  Remote user Auth想傳會被攔：challenge-response  **Security issue**  Replay登入上一個想燈的人(隨機產生驗證碼)  Trojan horse 讓你以為是官方，你送他密碼他超爽  Dos一直登入   |  | | --- | | **Ping flood**: ICMP/network bandwidth  缺點:來源會被clearly identified/自己會有reflection  Randomly spoof source: backcatter traffic | | **SYN spoofing**: TCP  victim一直收到ACK導致victim不能連線  SYN flood目標是數量取勝(network bandwidth)  **However, the flooding attacks are limited by a single system!!**  **防SYN spoof:** cookie/selective drop/table size(大) and timeout period(小) | | **DDoS** botnet zombie | | **Application-based**  SIP(session):INVITE  HTTP: Slowloris(blank line)/ HTTP flood下載很大的 | | **Reflector and Amplifier Attacks**  Q: Why Normal server systems?大多常看到的server  1. 容易易上手2. 難找到攻擊者  **Reflection**: UDP/TCP **Amplification**(broadcast):控制⼀堆zombie弄一堆request  一堆requests被廣播網路中，裡面的server們收到就response (**TCP不行廣播，UDP可以**)  **防flood**: block spoof/ ensure path back/limit一些rate | |  | | **Confidentiality:**  Data confidentiality: 我的資訊不能被知道  Privacy:你的資料被別人存  **Integrity:**  Data integrity: 可以改的人才可以改  System integrity: 系統能正常運作  **Availibilty:**可以用的人可以用  **Authenticity**:可信賴(訊息來源要正當)  **Accountablity**:責任性(你的動作可被追蹤)  Model for security: Hardware/Software/Data/Communication facilities  **Vulnerability**: weakness of system resources  **Corrupted**: loss of integrity  **Leaky**: loss of confidentiality  **Unavailable or very slow**: loss of availability  **Attack**: a threat that is carried out (threat action)  Passive: 沒改系統(純拿資訊 ex.竊聽)  Active: 有改(replay, masquerade, DoS)  Inside: by an authorized user  Outside: by an unauthorized user   |  |  | | --- | --- | | Unauthorized Disclosure  【Confidentiality】  Exposure資訊直接暴露  Interception攔截資訊Inference推斷:被猜到資訊  Intrusion侵入系統拿資訊 | Deception【Integrity】  Masquerade假裝自己是官方  Falsification用錯誤資訊欺騙官方  Repudiation不承認自己欺騙 | | Disruption【availability or system integrity】  Incapacitation癱瘓系統 Corruption亂改系統  Obstruction阻塞傳送 | Usurpation竄改  【system integrity】  Misappropriation侵吞別人系統資料DDoS  Misuse讓別人執行後不安全 |   **Fundamental Security Design Principles**  都用同一套系統性方法很好，但是別人也知道  Economy設計簡單/Fail-safe想你要甚麼不是你要甚麼/mediation檢查/Open design公開才會進步/separate of previlege規則不要太複雜/least common  Psychological acceptability不能安全到影響user/isolation隔離/encapsulation用oop壓縮/modualarity/layering/least astonish  **Attack surface**對誰攻擊 Network/software/human  Shallow layering+Large attack surface→high risk  Attack tree規劃攻擊路徑  ------------------------------------------------------------------------------Authtication驗證有效 / Authorize授權/ Audit審核  Access control   |  | | --- | | DAC(discretionary) 想幹嘛就幹嘛  →Subject可以改protection state  →查access matrix控制  →一個user一個row(protection domain) | | MAC(Mandatory) OS強制檢查access right | | RBAC(Role-based) 不同腳色不同權力  RBAC0: minimum functionality  RBAC1: RBAC0 + role hierarchies  RBAC2: RBAC0 + **constrains**  **Mutually exclusive** Role之間都不要有重疊的access  **Cardinality**限制role裡⾯面users的最⼤大數量量  **Prerequisite**有前提的role  RBAC3: RBAC0 + RBAC1 + RBAC2 | | ABAC(Attribute-base) 用屬性當條件  Distinguishable屬性要被定義好  Strength: flexibility and expressive power  Drawback: the performance impact  無限制數量的屬性fine-grained(大燕麥片是無限的) | | 傳統UNIX file access control用inode  No scalability: unwieldly and difficult to manage  現代UNIX:用ACL |   Access matrix常常太稀疏(sparse)  **ACL(Access control list)**對File來說清楚，對user爛  **Capability ticket**(ACL反過來)  Q: Have greater security problem than ACLs??  Tickets may be dispersed around the system  →OS holds all tickets on behalf of users  →An unforgeable token in the capability  **Authentication table**   |  | | --- | | **MIME mail format** | | **S/MIME有加密就是安全**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **plaintext** | **Bob’s Private k** | **1time session key** | **Alice’s Public key** | **Radix64 encode** | |  | **Digital signature** | **3DES** | **encrypt** |  |   **Enveloped data**→Encrypted content and associated keys/**Signed data**→Encoded message + signed digest/**Clear-signed data**→Cleartext message + encoded signed digest/**Signed and enveloped data**  →Nesting of signed and encrypted entities | |  | |  | |
|  |  |
|  |  |
| **Why Radix64?** ensure that the data remains intact without modification during transport. |  |
|  |  |
|  |  |