Protection against adversary

* **Secrecy**: can’t view
  + Prevent unauthorized access
* **Integrity**: can’t modify
  + Prevent unauthorized modification
  + **Authenticity**: prevent impersonation
* **Availability**: can’t deny

**Privacy**: right to control personal info

**Assurance:** procedures to ensure security policy

**Trust:** belief that system will do as expected

**Trusted Computing Base (TCB):** part of system assumed to function

**Accountability:** ability to attribute actions to individuals

**Access Control Lists (ACLs):** list to identify subjects with access

**Adversary:** someone against interests

* Adversary 🡪 Attacker when **acts**

**Motives:** curiosity, fame, money, national interest

**Resources:** time, money, training

**Security Models**

**Bell-LaPadula (BLP):** (secrecy)

* **No read up; No write down**

**Biba:** (integrity)

* **No read down; No write up**

**Buffer Overflow Defenses**

* Avoid unsafe functions
* Stack Canary
* Separate control stack
* ASLR
* Write/Exe (W^X)
* Control Flow Integrity (CFI)

**Attacking ASLR w/ W^X**

* Overwrite EIP with guessed location of usleep()
* Find base + offset of mapped region
* Overwrite EIP with address of ret
* Call system() with wget-ing script
* **Stack: random 24bits**
* **Mapped region: random 16bits**
* **Text/Data: random 16bits**

**Return-Oriented Programming**

* Examine gadgets, determine which registers can be read/written
* Create shellcode using existing code

**Password Authentication**

* **Checking Passwords:**
  + One-way function; store output
* **Dictionary Attack**
  + Generate hashes of all possible pw
* **Salting:** append random value

**Cryptography**

* **One-time Pads:** perfect secrecy;   
  each pad used once, else info leak
* **Stream Ciphers:** generate pseudorandom pad as long as plaintext
* **Block Ciphers:** encrypt/decrypt fixed-size messages (**secrecy**)
  + **DES:** key:56; block:64
  + **3DES:** key:168; block:64
  + **AES:** key:128,192,256; block:128
  + **Twofish:** key:128,192,256; block:128
* **Block Modes:**
  + **ECB:** Separately
  + **CBC:** XOR ciphertext block to next plaintext; start with random IV
  + **CTR:** block cipher 🡪 stream cipher
* **Hash functions (integrity)**
  + **MD5:** out:128; **known collisions**
  + **SHA1:** out:160; **weaknesses**
  + **SHA2:** out:224,256,384,512
  + **SHA3:** out: arbitrary
* **MAC: keyed hash w/ shared secret**
  + **HMAC:** MAC based on hash
  + **CMAC:** MAC based on block cipher
* **Asymmetric**: public/private key
  + **EIGamal:** hardness of discrete logs
  + **RSA:** hardness of factoring
  + **PKI:** public keys, trusted certificates

**Web Security**

* **Same Origin Policy**
  + **Origin:** <scheme, host, port>
* **Cookies:** way to manage session/auth
* **XSS:** include Javascript into page
  + Sanitize inputs/outputs
* **CSRF:** send request using another’s cookies
  + CSRF Tokens: random, per-form token
* **SQL Injections**: inject SQL code where page interprets; do stuff to database
  + Sanitize user input; prepared statements; escaping; minimize database privileges; store hash of sensitive data
* **HbbTV**: deliver web content through broadcast television stream (HTML/JS)
  + **Broadcast stream not authenticated**
  + Attacker able to jack into stream and inject malicious Javascript, etc.

**Principles**

* **Economy of mechanism:** Keep design as simple and small as possible
* **Fail-safe defaults:** Base access decisions on permission not exclusion
* **Complete mediation:** Every access to every object checked for authority
* **Open design:** not secret design
* **Separation of privilege:** protection mechanism requires two keys is more robust and flexible than with one key
* **Least privilege:** Every program and every user uses least set of privileges
* **Least common mechanism:** Minimize amount of common mechanism
* **Psychological acceptability:** Human interface designed for ease of use
* **Work factor:** Compare cost of attack with resources of potential attacker
* **Compromise recording:** keep record of information compromise

**Network Security**

* **TCP Connection Spoof**
  + Middleman has to complete TCP three-way handshake (SYN, SYN-ACK, ACK)
* **Address Resolution Protocol:** broadcast for unknown IP resolution
* **ARP Poisoning:** Impersonate and reply with false ARP
  + **Fix ARP tables; Port binding in switch**
* **DNS:** UDP domain name resolution
  + **UDP** has no authentication
  + **DNS Spoof:** spoof with off-path attack
* **Protect with SSL/TLS, DNSSEC**
  + Signed certificate with PKI

**Bitcoin & Electronic Payment System**

* Exchange using PKI, submit record on public ledger
* Utilize **blockchain** for system consensus of public ledger
* Finding blocks:
  + Output = SHA-256(Merkle Root + SHA-256(previous block) + Nonce)
  + Solution: output containing requisite number of leading 0 bits
  + Number of 0bits = **difficulty**
  + Difficulty adjusts