1 Theory

Properties

Confidentiality : prevention of unauthorized disclosure of info

Integrity : prevention of unauthorized modification of info

Availability : prevention of unauthorized denial of service

Others : authenticity, anonymity, non-repudiation

Threat model : adversary's capabilities and resources

Security policy : security properties that must hold

Vulnerability : weakness that could be exploited

Threat : feared event, goal of adversary we don't want materialized

Harm : the bad thing that happens, taken something protected

Vulnerability : weakness that could be exploited by adversaries

Trusted Computing Base TCB : every component of the system on which the security policy relies, trusted to operate correctly, ease verification, diminish the attack surface

1.1 Principles

Economy of mechanism : keep design as simple and small as possible

Fail-safe defaults : base access decisions on permission rather than exclusion

Complete mediation : every access to every object must be checked for authority

Open design : the design should not be secret, algorithms are pubic

Separation of privilege : no single accident, deception or breach of trust is sufficient to compromise the protected information

Least privilege : every program and every user of the system should operate using the least set of privileges necessary to complete the job

Least common mechanism : minimize the amount of mechanism common to more than one user and depended on by all users

Psychological acceptability : human interface designed for ease of use, users routinely and automatically apply protection mechanisms correctly

1.2 Access control

Access Control : Mechanism ensuring all accesses and actions on objects by principals are within the security policy, authorization

Reference monitor : centralized permission, program checks by calling the monitor, not follows least common principle

Discretionary Access Control DAC : object owners assign perms.

Mandatory Access Control MAC : central security policy assigns perms.

Access Control Matrix ACM : abstract representation of all permitted triplets (subject/principal, object/asset, access right/operation) within a system; no direct implementation, sparse (inefficient), error prone (hard to have global view)

Access Control Lists ACL : associate permissions to objects; store close/with resource, easy to determine who has access; difficult to check at runtime, difficult to remove/audit all rights of a user, difficult delegation

Capabilities : associate perms. to subjects; portable, easy to audit subjects, simple delegation; revoking perms. to one object is hard, authenticating the user, prevent sharing

Role Based Access Control RBAC : subjects are similar to each other, assign permissions to roles, assign roles to subjects, subject have permissions of their active role; can lead to role explosion (fine grained roles), problems with implement least privilege and separation of duty

Group Base Access Control GBAC : some perms. are always needed together, assign perms. to groups, assign subject to groups, subject have perms of all their groups

Ambient Authority : contextual subject, used by subject if they only specify operation and names of assets; no need to repeat subject (usability); least privilege harder to enforce

Confused deputy problem : with ambient authority, a privileged principal can be tricked to misuse its authority; solutions : re-implement access control in privileged process, let privileged process check authorization for user, capabilities can help

Linux permissions : read 4, write 2, execute/access 1; order owner, group, others; SUID (user) executes with permissions of file owner, SGID (group) executes with perms of group (on dir. group can access/execute new files), stiky bit (directory) only file owner can remove in the directory

1.2.1 Mandatory AC

Security Model : design pattern for specific security properties

Bell-La Padula model BLP : protecting confidentiality (no consideration on integrity or availability), subjects S, objects O, level of confidentiality; execute - cannot see or modify O but can run, read - see O but not modify it, append - cannot read but can attach new content, write - see O and add/modify content; state-based + single transition model (too low-level, not expressive); highest write is (low, {}), highest read (high, {all})

Security level : (label, set of categories), classification - total order of labels, categories - compartments of objects with common topic

Dominance relationship : (l_1, c_1) dominates (l_2, c_2) iff $l_1 \geq l_2$ and $c_2 \subseteq c_1$

Dominance lattice : dominates is transitive, partial order, top and bottom elements

Clearance level : clearance - max. security level S has been assigned, current security level - S can operate at lower security levels

Simple Security Property ss-property : if (S, O, r) is a current access, then level(S) dominates level(O), No Rea Up NRU

Star property *-property : if S has simultaneous (r, w) access to O_1 and (a, w) assess to O_2 then level(O_2) dominates level(O_1), No Write Down NWD

Discretionary Property ds-property : if an access (S, O, action) takes place it must be in ACM

Basic security theorem : initial state + transitions are secure $\Rightarrow$ every subsequent state is secure (regardless of inputs)

Covert channels : any channel allowing info. flows contrary to security policy; migration - isolation or add noise to communication

Declassification : remove classification label, cannot be made inherently safe, hard to rule out covert channels

1.2.2 Integrity Security Models

BIBA model strict : two operations (read, write), simple integrity (no read down NRD), *-integrity (no write up NWU), subject classification = label + category; dominance relationship (category superset and check label); highest read is (low, {}), highest write (high, {all})

BIBA low-water-mark for subjects : when accessing O the current level is lowered to lowest level btw. current-level(S) and level(O)

BIBA low-water-mark for objects : once O written by S assumed lowest level of O or S, allows for integrity violation detection

BIBA additional actions : simple invocation - allow S to invoke subjects with a label they dominate (+protect high integrity data from misuse, -level of output), controlled invocation - allow S to invoke subjects that dominate them (+prevents corruption of high integrity data, -hard to detect polluting info)

Sanitization : taking O with low integrity and lift them to high integrity, root of real-world security vulnerabilities, must follow fail-safe default (positively verify within valid set before elevation); white list (check all properties of good objects hold), do not blacklist (don't just check for bad objects/properties)

Integrity principles : separation of duties - require multiple principals to perform an operation, rotation of duties - allow a principal only a limited time on a particular role, secure logging - tamper evident log to recover from integrity failures

Chinese wall model : all objects have label denoting their origin, originators define conflict sets of labels, subjects are associated with history of their accesses to objects and their labels; subject can read an object if access does not allow an information flow btw. items with labels in same conflict set

1.3 Applied cryptography

Cryptography : ensure security properties (confidentiality, integrity), build secure functionality (authentication, denial of service, anonymous communication)

Confidentiality : info. cannot be accessed by unauthorized parties, send `e = Enc(k, m)`, decrypt `m = Dec(k, e)`

Integrity : info. cannot be modified by unauthorized parties

Encryption : plaintext to ciphertext (reversed is decryption), cannot be reversed without a **key** (as opposed to encoding)

One Time Pad OTP : key is random bits as long as `m`, encryption is XOR of m and k; key must never be reused : (m1 X k) X (m2 X k) = (m1 X m2); not used : long keys, key pre-shared (Moscow-Washington hotline), no integrity

Modern cryptography : security should not depend on secrecy (only on keys), algorithms are based on mathematically hard problems (prime factors, discrete logarithms), too complex to be executed by humans

1.3.1 Symmetric encryption

Symmetric : encryption and decryption share the same key (secret, known by both parties), key pre-shared once and reused (have a duration), stream ciphers & block ciphers, integrity with Message Authentication Code MAC

Initialization Vector IV : fixed-size input to iterative cryptographic primitives, no IV reuse under same key, is not secret

Stream ciphers : share `IV, e = stream(k, IV) XOR m`, stream is arbitrary length (not distinguishable from random if don't have key), `m = stream(k, IV) XOR e`; + speed (linear in time, constant in space) & low error propagation (errors in one bit do not affect subsequent symbols); - low diffusion (one plaintext symbols is one encrypted symbols) & susceptible to insertions or modifications (no integrity, hard to detect); Trivium (80 bit key), Salsa20 (128/25s bit key)

Block ciphers : decryption algorithm is inverse of encryption, `m = Dec(k, Enc(k, m))`, blocks same size as key (128/256 bits); + high diffusion & immunity to tampering (hard to insert without detection); - slow (block accumulated) & error propagation (errors affect several bits); AES (128/256 bit key)

Electronic Code Block ECB : encrypt/decrypt single blocks `e_i = Enc(m_i)` (problematic as two identical msg give same cipher)

Cipher Block Chaining CBC : add IV and propagate info. across blocks to introduce randomness; encryption `e_0 = IV` & `e_i = Enc(k, m_i XOR C_i-1)`, decryption `m_i = Dec(k, e_i) XOR C_i-1`; - IV incorrect full decryption is wrong & can't decrypt block alone

Counter Mode CTR : increasing nonce adds random. without dependencies between blocks, `e_i = Enc(k, nonce + i) XOR m_i`, can parallel en/decrypt

Message Authentication Codes MAC : hard to generate without key, fixed short sized key, send `(m, MAC(k,m))`, m cannot be tampered with, m comes from actual sender

CBC-MAC : `MAC(k, m) = e_n`, deterministic, output is final value, only secure if length of m known

Confidentiality + integrity : enc then plain MAC (no integrity of ciphertext, integrity of plaintext, reveal plaintext info.), mac then enc (no integrity of ciphertext, integrity of plaintext, no info on plaintext), enc then mac (full integrity and confidentiality)

1.4 Hash functions

Hash function : input `m`, output fixed short-length `h`

Properties : pre-image resistance (given `h`, hard to find `m`), second pre-image resistance (given `m`, difficult to find `n != m` s.t. `H(n) = H(m)`), collision resistance (hard to find any pair `m, n` such that `H(m) = H(n)`)

Usage : support digital signatures, build HMAC, password storage, file integrity, ...

HMAC : becarefull `HMAC != H(k || m)`

1.4.1 Asymmetric encryption

Asymmetric cryptography : each participant has two keys, one secret (decrypt, sign) and one public (encrypt, verify, stored in a public repository)

Confidentiality : send `e = Enc(PK0, m)`, read `m = Dec(SK0, e)`

Integrity : send `s = m, Sign(SK1, m)`, check `Verify(PK0, s)` (y/n)

Digital signatures : integrity of msg., authenticity of sender, non-repudiation, different key pairs than for confidentiality!

Public key infrastructure certificates : authority signs map btw. names and public keys,

authority signs maps btw. names and verification keys
Limitations : computationally costly, slow, not suitable to large amount of data
Digital signatures and hash : send `s = m, Sign(SK1, H(m))`, check `Verify(PK1, H(m), s)` (y/n), no need for primary image resistance as we send `m`
Hybrid encryption : establish a shared symmetric key using key transport, use the shared symmetric key to encrypt the rest of communication
Forward secrecy : secrecy of message in a session is kept if long term keys are compromised
Diffie-Hellman : shared public key parameters `p`, `g`, secret keys are `x` and `y` (random), `key = g^xy \% p`; discrete logarithm hardness; forward secrecy by deleting the secrets `x`, `y`

1.5 Authentication

Authentication : the process of verifying a claimed identity, bind actor to principal, before access control
Password : secret shared between user and system; problems : secure transfer (encryption, TLS), secure check (equal hash operations for accept and reject), secure storage (hash + salt), secure passwords; can be stolen, reuse
Replay attacks : send the encryption of the password that you got by listening, system sends a Nonce (challenge, public!) when wanting to log in
Hash salt : prevent attacker that has p + hash(p) of guesing p if presented with hash(p) again; use salt + hash(p + salt), with salt random, hash function slow
Prove : what you know, what you are (biometrics), what you have (smart card, secure tokens)
Biometrics : measurement and analysis of unique physical charateristis (figerprint, face, retina); data processing, false positive (wrong auth. accepted), false negative (true auth. rejected), config depend on application; -hard to keep secret, revocation difficult, private, not immutable
Token : 1 offline initialization (common seed and synchronize clocks), 2 optain number from seed with token, `n = now_start / interval`, compute `v = f^n(seed)`, 5 send result to server, server also compute `v` and check they are the same; atacker observe `v_n` cannot guess `v_{n+1}`
Machines : have secret key, digital signatures

2 Attacks and defenses

Threat modeling : process to identify potential threats and unprotected resources (with goal of implement security mechanisms); attack trees (goal is root, leafs are weak resources), STRIDE, PASTA (threats within business model)

STRIDE : Spoofing (authenticity), Tampering (integrity), Repudiation (non-reputability), Information disclosure (confidentiality), Denial of service (availability), Elevation of privilege (authorization)

Insecure Interaction Between Components CWE I : data sent and received between programs
Common Weaknessses Enumeration CWE : database of software errors leading to vulnerabilities
OS Command injection CWE-78 : user-input executed as part of a system command on server; sol. sanitization
Cross-site scripting XSS CWE-79 : take user-input and insert it into WEB html (extract data/cookie, execute commands); sol. sanitization
Cross-site Request Forgery CSRF CWE-352 : Use post request from maliciously crafted form to server in order to execute action with users cookie without them agreeing (confused deputy, ambient authority); sol. : confirm actual origin of authority and request (HTTP referrer/origin field), add a challenge (stored in cookie, sent in form), re-authenticate for every action
Same Origin Policy SOP : script cannot access data from another origin (same protocol, same host, same port); another site cannot read cookies from other sites, not help CSRF

Risky Resource Management CWE II : manage creation, usage, transfer, destruction of system resources (buffer overflow, TCB control); download code without integrity check, Inclusion of functionality from untrusted
Porous defenses CWE III : defensive techniques misused abused or ignored (missing checks, partial mechanisms, hard-coded credentials)

2.1 Memory attacks

Memory Corruption : Unintended modification of memory location due to missing or faulty safety check
Temporal error : access memory that was reserved to the program later when it is not reserved anymore
Spacial error : access memory that was not reserved to the program
Uncontrolled Format String CWE-134 : use format strings like `%s,%d,%4$p` without actual value, reads data from the stack; sol. : `printf("%s", var)`
Code injection : use memory corruption to write code in the stack frame
Data Execution Prevention DEP : implemented at hardware level, enforces code integrity on page granularity, executable bit, either write or execute; -no self modifying code support
Code reuse : find addresses of gadgets, use memory corruption to redirect control flow to gadget chain
Mitigations : stop unknown vulnerabilities, exploitation harder but not impossible
Address Space Layout Randomization ASLR : prevent the attack from reaching a target address, randomizes locations of code and data regions; - prone to information leaks, some regions remain static, bad performances
Stack canaries : canary value between vulnerable buffer and return address, stop execution if the value changes unexpectedly; - prone to information leaks, no protection against targeted read writes

Testing : error - deviation between actual observed behavior and specified expected behavior, practical testing is limited by state explosion
Bugs : Testing can only show the presence of bugs, never their absence.
Manual testing : designed by human, code review, heuristic test cases; Exhaustive, Functional, Random, Structural (coverage)

Automated testing : algorithmic, run program and find bugs, check enforced properties; static analysis, symbolic analysis (symbols and branch analysis, not scale), dynamic analysis (fuzzing, run program)
Coverage : statement coverage - what % of statements have executed; branch coverage - what % of possible paths were taken
Fuzzing : Random testing, mutate inputs to improve coverage, use coverage feedback to mutate input
Sanitization : assertions, segmentation faults, division by zero, uncaught exception, mitigation triggering termination
Address Sanitizer ASan : detects memory errors, red zones around objects, check on trigger events; detects - Out-of-bounds accesses, Use-after-free, Use-after-return, Use-after-scope, Double-free, invalid free, Memory leaks; -very slow
Undefined behavior Sanitizer UBSan : detects undefined behavior - Unsigned/misaligned pointers, Signed integer overflow, Conversion between floating point types leading to overflow; used in production

2.2 Network security

Naming security : association between lower and higher level names (network address, surname); integrity, authentication, availability
Routing security : route over network not influenced by adversary; integrity, authentication, availability, authorization

Session security : messages within session cannot be modified (ordering, add, remove); integrity, authentication
Content security : content of messages not readable or influenced by adversaries; confidentiality, integrity

Ethernet : local area network LAN, unique 48 bit Medium Access Code MAC address
Internet Protocol IP : hosts communicate using IP, each machine has IP address (4 bytes IPv4)
Address routing protocol ARP : "translation" between IP and MAC addresses; if not known broadcast ARP request to query MAC for target IP, ARP reply responds with MAC address corresponding to IP

ARP spoofing : vulnerable to impersonation (if nobody checks), man in the middle MITM, DOS; same with DNS, IP, Ethernet, no network protocol was designed with security in mind; sol. : use static read-only entries in ARP cache host, ARP spoofing detection and prevention (check if on IP has more than one MAC, certify requests by cross-checking)
DNS spoofing : Cache poisoning - corrupt DNS resolver with fake pairs (IP, domain); DNS Hijacking - corrupt DNS responses MITM with fake pairs; can achieve DOS, redirection - reroute clients to malicious host (malware, monitoring); sol. : DNS Security Extensions DNSSEC (origin authentication, digitally signed by authoritative name server) not encrypted so no confidentiality, DNS-over-HTTPS DoH
BGP spoofing : inject false low-cost routes to redirect portions of traffic to themselves, routing information propagates to routing tables (redirection, surveillance, injection, censorship); sol. : filtering help alleviating, no authority to guarantee correctness of routes, BGPsec each AS given certificate links it verification key to its IP blocks, updates accepted only if they are signed, delegation is possible

IP Security IPSec : Authentication header AH (authentication and integrity HMAC, protect replay attacks), Encapsulating Security Payload ESP confidentiality; Transport mode (IP packet payload protection, sent with original headers) Proxy, Tunnel mode (protects whole packet) VPN
IP limitations : no reliability, no congestion/flow control, no sessions, no multiplexing
TCP hijacking : MITM can observe communication and intercept/inject packets
TCP 3-way handshake : if adversary guess seq numbers they can hijack and insert packets (weak RNG or observation if sent in clear)

Transport Layer Security TLS : cryptographic protocols above TCP/IP (middlelayer), confidentiality (symmetric encryption), authentication (public key cryptography), integrity (MAC and signatures), forward secrecy (learning a secret does not reveal anything about the past)
TLS handshake : client - ClientHello, Version, CipherSuites (encryption, hash), SessionID, Challenge; server - ServerHello, ChosenCipher, ServerCertificate (PKI certificate authenticate server), ServerKeyExchange, ClientCertificateReq, Challenge; client - ClientCertificate, ClientKeyExchange, ChangeCipherSec (from now on, encrypt everything) + ClientFinish (encrypted); server - ChangeCipherSec + ServerFinish

Denial of Service DoS : prevent legitimate users from accessing a service; crash victim (software flaw), exhaust victim's resources (network bandwidth or kernel/application)
SYN flood : send TCP SYN packets with bogus src address, TCB entries exist until timeout, kernel limits # of TCB; sol. : minimize state before you are athenticated, don't create TCB (a few bytes per connection), push state to client (SYN cookies) cryptographically client state
Network Address Translation NAT : router that maintains routing tables (internal IP, port) to (external IP, port), external entity cannot route into the NAT
Network Firewalls : network router that connects an internal network to external (public) network, mediates all traffic, makes AC decisions (allow, deny traversal), prevent dangerous flow; - full mediation is slow (observation is cheaper), authenticate all principals, correctness of data
Defense in depth De-Militarized Zone DMZ : WAN outside, DMZ public services, LAN internal users; relies on firewall; LAN can acces DMZ and WAN, DMZ can access WAN, flows in other direction restricted monitored authenticated; safe internal resources

3 Privacy

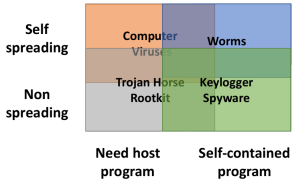
Privacy : is a security property; denying privacy to some is denying to all
For individuals : protection against profiling, manipulation, crime, identity theft

For companies : protection of trade secrets, business strategy, internal operations, access to patents
For governments/military : protection of national secrets, confidentiality of law enforcements investigations, political negotiations

Privacy Enhancing Technologies PETs : social adversary - problem defined by Users, do not surprise the user, limitations (protect from other user, trusted service provider); institutional privacy - personal data should not be collected, data should be secured, compliance with data protection rules (informed consent, purpose limitation), auditability and accountability, limitations (trusted service provider, limited scope); anti-surveillance privacy - defined by security experts, adversary is anybody, minimize need to trust, limitations (usability problems, performance, lack incentives)
End to End Encryption : confidentiality, forward secrecy using ephemeral keys, problem with traffic analysis (metadata is also sensitive), address where data is stored may reveal information about content, address where action happens may reveal information about action/user
Traffic analysis : process of intercepting and analysing messages from communication patterns
Targets : hackers, journalist, whistle-blower, human rights activist, business executive, military intelligence personnel, abuse victims
Anonymous communications : bitwise unlinkability (input and outputs appear different, use crypto); re-packetizing, re-schedule, re-route (destroy patterns, load balancing, distribute trust)
Tor network : onion routing; select a path, prepare the circuit (authenticated Diffie Hellman to get symmetric key), send stream; provides privacy as long as adversary cannot see both edges
Low latency : stream based, web browsing, instant messaging, streaming (path fixed per stream); cannot resist Global adversary
High latency : MSG-based, email, voting, bitcoin, (path vary every message); global adversary resistance (latency cost)
Application layer : anonymous credentials, attribute based credentials, singed by trusted issuer; anonymous users, unlinkable across contexts

4 Malware

Malware : malicious-software, intentionally written, perform some unwanted activity; 60% trojans, 20% viruses, 7% worms, ...
Rise : homogeneous computing base (windows, android), clueless user base, unprecedented connectivity (remote, distributed attacks), malicious code is profitable (sold/used to make money)



Virus : infects programs, monitor steals or destroy, modify programs to include copy of themselves, cannot survive without host, allowed to do anything host program is permitted to do, replicates to infect other content or machines (network or hardware); file infection, macro infection (word, excel), boot infection
Virus defenses : signature-based detection (byte-level or instruction level signatures matching virus, hash of malicious programs), sandboxing (run untrusted applications in restricted environment, VMs)
Worm : self-replicating self-contain (no host), use network to send copies, autonomous spread (email, network)
Worm defenses : host-level - diversity (heterogeneous systems, clash with economy of mechanism), anti-virus, stack protection, remote exploitation protection; network-level - limit outgoing connections (limits spreading), personal firewall, intrusion detection systems
Intrusion detection system : host (local malware), network (monitor all traffic); signature (known patterns) + low false alarms - expensive (up-to-date signatures), anomaly (behavior) + adapt to new attacks - high false alarms
Trojan Horse : appears to perform correctly but also undisclosed malicious activities, users explicitly run program (defense train users), cannot replicate; spy/steal, remote access (backdoor), further attacks (email relay, spammers), damage routines
Rootkit : adversary code deep within TCB (modify the OS, kernel, system programs), installed after system compromised, hard to detect, adversary can return later; defense - integrity checkers on user/kernel level
Backdoor : hidden functionality, adversary can bypass some security; audit program source (compiler can introduce backdoors!); suspect all programs
Botnets : multiple (millions) compromised hosts (bots, zombies), controlled by single entity; Command & Control system C&C (single point of failure, violate least common mechanism); pear to pear P2P, difficult management (hybrid exists)
Monetizing botnets : rental, DDoS extortion, bulk traffic selling (boost visits), click fraud (simulate clicks on ads), distribute ransomware, advertise products (leave comments), bitcoin mining
Botnets defense : Attack C&C (hijack/poison DNS to route traffic to black hole), honeypots (study attacker behavior in controlled environments)
Other malware : rabbit (replicate to exhaust resources), logic (time) bomb (triggers action when condition (time) occurs), dropper (drops other malicious code), toolkit (assemble malicious code, not malicious itself), scareware (false warning of malicious attack)