Distributed Operating Systems

Security in Distributed Systems

dr inż. Adam Kozakiewicz

akozakie@ia.pw.edu.pl

Institute of Control and Information Engineering
Warsaw University of Technology

Security Terminology

vulnerability a fault in the system enabling an attacker to cause inappropriate (in the worst case – malicious) behavior

threat something that threatens the system

compromise the effect of a successful attack

exploit a program exploiting a vulnerability

payload a program executing the actions planned as the goal of the attack

rootkit a (set of) program(s) hiding the attacker's presence

malware a generic name for malicious programs, e.g.:

- √ backdoor – a secret part of the program, enabling unauthorized access
- √ trojan – an otherwise benign program with an undocumented, secret side-effect
- √ logic/time bomb – a fragment of code executed when a given condition is satisfied
- √ virus – a program propagating itself by adding its code to other programs
- √ worm – a program propagating itself actively through the network
- √ bacteria – a program multiplying itself locally to exhaust the system's resources

Life Would Be Easier If Anybody Remembered That...

- √ Security of the software is immaterial if physical security is lacking! If an attacker has a direct, unsupervised, physical contact with the machine, then any protections are at most nuisances.
- √ Threats are usually not attacks. However you count the damages, most of them are caused by human mistakes.
- √ External threats are the easiest to protect against – most of the really effective attacks are the work of the affected company's own employees.
- √ The weakest link in the chain of security are always people. The most successful attacks are preceded by simple tricks – rummaging in the trash, social engineering, observation.
  - ⋆ users are usually naïve and eager to help,
  - ⋆ users have work to do, security procedures are an obstacle,
  - ⋆ e.g. strict rules for choosing passwords are very effective against dictionary attacks, but automagically cause post-it notes to appear on many monitors and keyboards...

Types of External Attacks

- √ attacks on connetions
  - ★ sniffing – watching packets in existing connections
  - ★ spoofing – impersonating an authorized machine in communication with the victim (mostly IP spoofing)
  - ★ smurf – impersonating the victim in communication with other machines
  - ★ land – impersonating the victim in communication with…the victim
- √ attacks on machines/processes
  - ★ information theft
  - ★ taking over the machine (zombies, botnets – e.g. for DDoS attacks)
  - ★ vandalism
- √ denial of service
  - ★ direct – DoS – Denial of Service, DDoS – Distributed Denial of Service
  - ★ indirect – any type of attack agains the network infrastructure

Security of Information Systems

Basic elements of information system's security:

- √ isolation and minimization of services
- √ access control
- √ cryptography
  - ⋆ ciphers
  - ⋆ authentication
- √ elimination of vulnerabilities

Isolation and Minimization of Services

Motto: You can't attack something that isn't there!

- √ No external attacks work on a system physically disconnected from the Internet.
- √ **Firewall** – traffic filtering device or software
  - ⋆ drop traffic attempting to access services that should not be offered by the protected servers
  - ⋆ a good firewall policy is to block everything and treat any required traffic as an exception
- √ A secure machine should run only a minimal set of absolutely necessary programs
  - ⋆ If a service/program can be uninstalled – do it!
  - ⋆ If you can't – disable it!
  - ⋆ If you can't – disable access to it!
  - ⋆ If you can't – configure it well (use best practices for that application)!

DMZ – De-Militarized Zone

Trusted zone

The internal network of a system, connections within this zone are assumed to be safe (that doesn't mean you shouldn't filter them!)

Demilitarized zone

A subnetwork for servers offering services to any machines outside the trusted zone. Computers in this zone should not be trusted.

- √ any machine accepting any connections from „outside" (from the Internet) can be compromised and should not be trusted,
- √ no such machines are allowed in the trusted zone,
- √ a separate demilitarized zone is set up for such machines,
- √ a firewall separates the DMZ from the world, allowing only connections to officialy available services,
- √ a separate firewall protects the trusted zone allowing only a very limited set of connections from the DMZ (none, if possible).

Security Policy

- **Security policy** determines the right of each user to perform specific actions on specific resources.
- Minimal or maximal privilege policy:
  - minimal – privileges for actions **necessary** to perform a role
  - maximal – privileges for actions **possibly applicable** in a given role
- Access control (default behavior):
  - **open** – if an action is not explicitly forbidden, it is allowed
  - **closed** – if an action is not explicitly allowed, it is forbidden
- Granularity of access control

Access Control Models (1)

Access control

A logical or physical control mechanism protecting the system from **unauthorized entry** or **use**. Controls:

- √ *which* **users** (machines, processes, employees) have access,
- √ *to which* **resources** of a computer system,
- √ *what kind of* access,
- √ *how* can they use **shared data**.

Access control works on many levels – in the application layer, middleware, operating system, hardware...

Access Control Models (2)

- √ **discretionary access control, DAC**
  - ⋆ the owner of each resource specifies the access rights,
  - ⋆ tradeoff between scalability and granularity, with low scalability anyway.
- √ **mandatory access control, MAC**
  - ⋆ *centralized* rights management,
  - ⋆ **security level** assigned to every resource,
  - ⋆ **clearance level** assigned to each user,
  - ⋆ pretty scalable, but questionable granularity.
- √ **role based access control, RBAC**
  - ⋆ access rights are assigned to **roles**,
  - ⋆ a user may have more than one role,
  - ⋆ simple user management, lowers administrative costs.

Role Based Access Control

Model	Hierarchical roles	Role constraints
$RBAC_0$	No	No
$RBAC_1$	Yes	No
$RBAC_2$	No	Yes
$RBAC_3$	Yes	Yes

Hierarchical roles roles may include other roles

- √ simplified management,

Role constraints constraints on possible combinations of roles or simultanous application of assigned roles

- √ Example: employee of a bank may also be its client, but may not assume both roles in one transaction (e.g. accept his own loan application).

Trust Management

Example: KeyNote (RFC 2704).
A trust management system specifies:

- √ language for describing of actions
- √ mechanism for identifying principals
- √ language for specifying application policies
- √ language for specifying credentials
- √ compliance checker

Assertion – specifies the authorization for a given action
Monotonic asserts – adding an assertion can only increase the set of allowed actions, not reduce it.

- √ Makes management easier, but how to revoke rights?

Role Based Trust Management

- √ Role based trust management (RT) combines the features of RBAC and trust management.
- √ Corresponds most closely to attribute-based access control (ABAC).
- √ Properties of RT:
  - ⋆ localized authority over roles
  - ⋆ delegation in role definition
  - ⋆ linked roles
  - ⋆ parametrized roles
  - ⋆ manifold roles

Attribute Based Access Control (1)

An attribute based access control system specifies:

- √ decentralized attributes
- √ delegation of attribute authority
- √ inference of attributes
- √ attribute fields
- √ attribute-based delegation of attribute authority

Attribute Based Access Control (2)

Example 1 (RT_0):
An application A made available by division D of company C may be used by employees of the company having a security clearance certificate issued by the security division S. Alice may use the application:

$$D.authorizedA \leftarrow D.known \cap D.clearance, \quad D.known \leftarrow C.employee$$

$$D.clearance \leftarrow S.clearance, \quad S.clearance \leftarrow Alice, \quad C.employee \leftarrow Alice$$

Example 2 (RT_1):
An action H may only be performed by an employee with at least 2 years seniority, irrespective of the position in the company.

$$Company.authorizedH \leftarrow Company.employee(?, ?seniority : [2..50])$$

employee is a parametrized role with two parameters – position and seniority.

Attribute Based Access Control (3)

Example 3 (RT_1):
Application allows entering the annual evaluation of an emplyee only if the user is that employee's direct manager.

$$Company.evaluatorOf(?Y) \leftarrow Company.managerOf(?Y)$$

Example 4 (Non-monotonic RT):
A program may not be considered tested if only his author tested it. Who can enter valid testing reports?

$$Company.verification \leftarrow Company.tester \ominus Company.programmer$$

$$Company.tester \leftarrow Alice, \quad Company.tester \leftarrow Bob, \quad Company.programmer \leftarrow Alice$$

Only Bob may test Alice's code.

Basics of Cryptography

The goals for cryptography:

Secrecy (confidentiality) – a message cannot be read without a key

Authentication – confirmation of identity

Integrity – any modification of the message will be detected

Undeniability – the author of the message cannot deny sending it

Some applications: digital signature, blind signature, information sharing, digital elections, digital banknote...
These goals are achieved through application of cryptographic **algorithms** and **protocols**.
Note: cryptography is extremely sensitive to incompetent implementation. Every step of each algorithm is there for a reason, often counterintuitive. Even a minimal change, even one meant to increase the security, may make an algorithm much easier to break and therefore useless.

Cryptography – Terminology

cryptography „secret writing”, in other words ciphers, etc..

cryptoanalysis breaking of the products of cryptographer's research

cryptology cryptography + cryptoanalysis, but often called cryptography anyway

plain text what we want to hide

cryptogram what we get after applying a cipher – unreadable message

cipher the mechanism converting plain text into a cryptogram and vice versa

key parameter of the cipher, the value differentiating its effects – without knowing the key the cryptogram cannot be deciphered

Hash Function

One-way hash function

A function $H(M)$ of argument (message) M with the following properties:

- √ the result h has a constant length, irrespective of the length of M,
- √ computing h for a given M is easy,
- √ for a given $h = H(M)$ finding M' such that $H(M') = h$ is an unrealistically complex task.

These properties make the hash function a perfect part of a digital signature mechanism and a very good error detection tool in communication.

For some application **collision resistance** is also useful. This means that it is difficult to find a pair M and M' such that $H(M) = H(M')$.

Examples: MD5, SHA-0, SHA-1, SHA-256, SHA-384, SHA-512.
Current standards have already been broken or at least weakened (collisions can be generated) – the situation is still not clear.

The Perfect Cipher – One-Time-Pad

- √ Cannot be broken, extremely fast, very simple to implement... no, it's not a fantasy, this cipher really exists!
- √ **Single-use key**, generated using a truly random source, known only to the sender and the receiver, at least as long as the message.
- √ Algorithm: A simple bit-wise XOR:

$$\begin{aligned} cryptogram &= plaintext \otimes key \\ plaintext &= cryptogram \otimes key \end{aligned}$$

- √ Usually not a practical solution due to problems with key management.
- √ Watch for „snake-oil vendors"! „Similar/almost like one-time-pad" means *definitely not one-time-pad*. Not even one bit of the key may be reused and the key must be truly random!

Symmetric Ciphers

- √ Both sides of the transmission have a common, identical, **secret** key, transmitted using a **secure** channel,
- √ Data blocks are decrypted using an inverse of the encryption algorithm,
- √ Quality of the cipher is determined by the structure of the algorithm, length of the data block encrypted in each step and length of the key,
- √ Fast, reliable, secure, effective, efficient... just great,
- √ Key management is a problem:
  - ⋆ the keys must be communicated to their users in a secure way,
  - ⋆ for point-to-point communication the number of keys grows exponentially.
- √ Examples: DES, 3DES, AES (Rijndael), Blowfish, Twofish, IDEA, RC5...

Asymmetric Ciphers

- √ **Two** keys are used during transmission.
- √ Decryption **is not** an inverse operation to encryption.
- √ Much slower, most of these algorithms require very long keys to be secure.
- √ Each user receives two keys – private and public. Depending on the application, one of them is used to create a cryptogram, the other – to decipher it. Some algorithms actually work both ways.
- √ The algorithms are based on the difficulty of some mathematical problems, so in theory they can be broken by finding a good algorithm to solve these problems, e.g. a fast factorization or discrete logarithm algorithm.
- √ Even so, these algorithms caused a real revolution in cryptology!
- √ Examples: RSA, El-Gamal algorithm, eliptic curve algorithms.

Public Key Infrastructure (1)

Problem How to obtain a public key of someone we never contacted before without risking compromise?

Solution a confirmation by a trusted third person, in other words a signature on the key. Possible implementations:

- √ *trust network* – users hold their own keys and publish a set of signed keys, which they consider trusted.

  A key trusted by several trusted users can be trusted (more elastic approaches possible).
- √ centralized key management – not scalable.
- √ hierarchical approach – **certificate chains**.

Public Key Infrastructure (2)

Certification Authority (CA)

Institution responsible for signing (issuing) certificates, containing public keys of the users of the system.

The de facto (and de jure in some applications) standard is X.509v3.

Every user should have the certificate of the CA, used to verify the signature on other certificates.

Note The signature on the certificate only confirms, that the person using the private key associated with the certificate was – in the opinion of the CA – indeed the person, whose data appear in the certificate. In no way does this guarantee that this person will not lie to you, steal from you, kill you or blow the planet to pieces.

X.509v3 Certificates

Some important information contained in X.509v3 certificates:

- √ the owner's public key (of course),
- √ issue date,
- √ expire date,
- √ owner's identification (X.500)
- √ issuer's identification
- √ extentions (v3 and Netscape) specifying valid uses of the certificate
- √ issuer's signature (required, but self-signed certificates are possible)

Certificates can be revoked:

- √ certificate revocation lists (CRL), checked (rarely...) using OCSP (Online Certificate Status Protocol)
- √ expire date – limits the length of CRLs

Key Negotiation (1)

Problem 1: Asumetric ciphers are too slow to be used for encryption of connections. **Problem 2:** Symetric ciphers require a key to be known to the sender, the receiver and noone else.
Solution: Key negotiation (exchange) protocols. A simple example:

1. A sends a random number to B, encrypted using the public key of B,
2. B decrypts the number using the private key,
3. B sends a random number to A enncrypted using the public key of A,
4. A decrypts it,
5. A and B join both numbers and optionally hash it, generating a key,
6. A and B create a connection using symmetric encryption with the generated key.

Key Negotiation (2)

Advantages of the generated key:

- √ Both participants put in the key random numbers they generated, so they know that man-in-the-middle attack using packets from previous connections did not happen.
- √ This algorithm only works if the private keys are secret, so that noone may impersonate them.
- √ All information passed between A and B is encrypted.

The SSL/TLS Protocols

- √ Product of Netscape: SSL (ang. *Secure Socket Layer*), versions 1, 2 and 3, de facto standard.
- √ Accepted as IETF standard, but renamed to TLS (ang. *Transport Layer Security*), versions 1.0 and 1.1.
- √ Protocols for encryption of TCP connections.
- √ Provides authorization, secrecy, integrity, but not undeniability (workarounds are possible).
- √ A wide selection of ciphers available.
- √ Can be (but rarely is) used to tunnel any protocol, in practice usually protects HTTP (HTTPS = HTTP + SSL/TLS) or e-mail (SMTP, POP3, IMAP).
- √ **WARNING:** SSL2 is vulnerable to an attack called reverse key negotiation, do **not** use it – use at least SSL3!

The SSH Protocol

- √ A secure replacement for telnet, written by Tatu Ylönena, later extended to allow file transfers (SCP/SFTP) and tunneling of other protocols; versions 1 (in fact 1.5) and 2, both still in use.
- √ Both users and computers are authenticated, but no certificates are used.
- √ Ciphers: RSA, sessions use DES, 3DES, Blowfish.
- √ The server uses two pairs of asymmetric keys, used to negotiate the symmetric key:
  - ⋆ constant, generated during installation – authenticates the server,
  - ⋆ temporary, generated on server startup and often changed – protects sniffed communication from decryption if the server is later compromised.
- √ User authentication depends on configuration: public key of the user or machine, password, or even automatic accept.
- √ SSH and SSL are two different, completely separate tools!

IPSec

- √ Obligatory part of IPv6, optional part of IPv4,
- √ A protocol suite:
  - ⋆ *Internet Key Exchange (IKE)* – key exchange protocol (what else?)
  - ⋆ *Encapsulating Security Payload (ESP)* – protocol protecting connections, ensures secrecy, authentication and integrity,
  - ⋆ *Authentication Header (AH)* – another protocol protecting connections, ensures only authentication and integrity.
- √ Modes:
  - ⋆ *tunelling* – connected nodes protect connections going through them, normal IP outside.
  - ⋆ *transport* – connections set up from end to end as IPSec, intermediate nodes forward already secured packets.
- √ Usually used to create secure VLANs.
- √ Transport mode not very popular – no sufficiently large-scale, universally accepted PKI is available.

Cryptography – What We Ignored

What we might (should?) have discussed (but we won't!):

- √ Kerberos – cryptographic infrastructure for an organization (MIT),
- √ PEM (ang. *Privacy Enhanced Mail*), PGP (ang. *Pretty Good Privacy*), GPG (ang. *GNU Privacy Guard*) – cryptographic infrastructure for Joe Sixpack
- √ problem of random number generation,
- √ most cryptographic algorithms and protocols and many other things...

Always remember:

- √ Good encryption comes only from a secret key. A secret algorithm only means that it wasn't verified by independent researchers.
- √ *Rubber hose cryptoanalysis* is always the simplest approach for an attacker. If you can e.g. steal the key, cryptography is useless.

Elimination of Vulnerabilities

Every system should be constantly updated (security updates only, of course).

Most vulnerabilities are an effect of insufficient verification of input:

- √ buffer overflows.
- √ SQL injection – input data passed directly to SQL queries, without proper masking of control characters, e.g.:
  `SELECT * FROM transactions WHERE trans_id=$transId;`
  Put `1132342; drop database` as transaction ID – what will happen?
  With a good configuration – not much, but this was just an example.
- √ other injection attacks – far too often you can see in the address bar of your browser pretty things like `?admin=no`, or `?file=/home/dumbadmin/thefile.html`. Tempting, very tempting!

Any data generated or collected outside the system should never be trusted, even if we wrote the client's code! Code can be modified, it's results can be replaced. Bad input can always be injected – break that needle, write the servers with iron skin!