Operating System Security

OS Security

- OS is the lowest layer of software that's visible to users
- OS is very close to the hardware and often have complete hardware access
- OS isn't protected, machine isn't protected
- Flaws in the OS generally compromise security at higher levels

  Security Importance

  - OS controls access to app memory, scheduling of the processor, nd ensures that users receive the resources they ask for
  - OS isn't doing things securely, then other operations can go wrong
    - Memory management, persistent storage devices, running processes
  - Almost all other security systems must assume a secure OS at the bottom

  Security

  - A policy
  - Ex. No unauthorized user may access this file

  Protection

  - A mechanism which implements security policies
  - Ex. System checks user identity against access permissions

  Vulnerabilities and Exploits

  - Vulnerability is a weakness that can allow an attacker to cause problems
    - Not all vulnerabilities can cause problems
    - Most vulnerabilities aren't exploited
  - An exploit is an actual incident of taking advantage of a vulnerability

  Trust

  - Trust the OS since it controls the hardware and memory

Authentication

- Authentication: know whos asking
- Authorization: need to check that the party should be allowed to do it
- Security policies tend to allow some parties to do something and not others
- We need to know whos doing the asking
- Ex. ID by recognition, credentials, knowledge

  Identities in OS

  - Rely on user ID, which uniquely identifies some user
  - Process run on his behalf and inherit his ID
  - A process which belongs to a user has some of their privileges

  Bootstrapping OS Authentication

  - Processes inherit their user IDs

- We have to create a process
- We have to create a process belonging to a new user
- We can’t inherit that identity

Passwords

- Authenticate the user by what he knows (password)
- System must be able to check that the password was correct using a hash
- If correct tie the user ID to a new command shell

Problems with Passwords

- Have to unguessable
    - Easy for people to remember
- If networks connect remote devices to computers, then susceptible to password sniffers
- Brute force attacks

Proper passwords

- Should be long
- Should be unguessable
- Shouldn’t written down
- Shouldnt be shared
- Hard to achieve all at the same time

Challenge/Response Systems

- Authentication by what questions you can answer correctly
- System asks the user to provide some information

Hardware Challenge/Response Time

- Challenge is sent to a hardware device belonging to an appropriate user
- Sometimes having the device is enough
- Sometimes the device performs secret function of the challenge
    - Smart cards

Problems with Challenge/Response

- Usually there are too few unique and secret challenge and response pairs, so the response can be found by attackers
- If you soe the device, you can’t get it in anymore
- Can be susceptible to network sniffing

Biometric Authentication

- Authentication based on what you are
- Measure the physical attributes like finger print or voice patterns
- Convert to binary representation, then check for a close match

Problems with Biometrics

- Requires very special hardware
- Many physical characteristics which vary too much for practical use
- Generally not helpful for authenticating programs or roles
- Can still be exploited if done across a network

Errors in Biometric Authentication

- False Positives
    - Incorrectly identified person 1 as person 2

- False Negatives
    - Even though I'm authorized, the biometrics doesn't work
    - Variability results in the scanner being picky

Biometrics and Remote Authentication

- Biometric reading is a bit pattern, which is sent over a network
- An attacker can obtain a copy and use it
- Biometrics over networks need high security between the biometric and checking device

Multi Factor Authentication

- Rely on two separate authentication methods
    - Password and a text message
- If done well, compensates for the other methods downsides
- Currently the preferred method

Access Control List (ACL)

- For each protected object, maintain a single list
    - Managed by the OS
- Each list entry specifies who can access the object
- When something requests access to a object, check the access control list

UNIX ACL

- Owner, group, other
- 3 modes: Read, write, execute

ACL Advantages

- Easy to figure out who can access a resource
- Easy to revoke or change access permissions

ACL Disadvantages

- Hard to figure out what a subject can access
- Changing access rights requires getting to the object

Capabilities

- Each entity keeps a set of data items that specify his allowable accesses
    - Similar to set of keys that an entity keeps
- To access an object, the proper capability is presented
- Having the capability for an object implies that access is allowed

Properties of Capabilities

- A data structure (collection of bits)
- Simply having the capability grants access
- Cannot be foreable
    - Don't let the user processes have them
    - Store them in the OS

Advantages of Capabilities

- Easy to determine what objects a subject can access
- Faster than ACLs
- Easy model to transfer privileges

Disadvantages of Capabilities

- Hard to determine who can access an object
- Requires extra mechanism to allow revocation
- In network environment, need cryptographic methods to prevent forgery

OS Use of Access Control

- Operating systems often use both ACLs and capabilities
    - Sometimes both used to verify the same resource
- Creates a file descriptor with a particular set of access rights
- Descriptor is a capability

Enforcing Access in an OS

- Protected resources must be inaccessible
    - Hardware protection must be used
    - Only the OS can allocate protected resources
- Requests must be made to the OS through syscalls
    - OS cosults access control policy data
- Access may be granted directly
    - Resource manager maps resource into process
- Access may be granted indirectly
    - Resource manager returns a capability to process, which can then be used by the app to access the resource

Cryptography

- Encoding a string of bits to make it hard to read
- Described in terms of sending a message
- Sender is S, receiver is R
- Encryption
    - The process of making message unreadable by anyone but R
- Decryption
    - The process of making the encrypted message readable by R
- Cryptosystem
    - A system performing these transformations
    - Cipher: rules for transformation

Plaintext and Ciphertext

- Plaintext (P): The original form of the message
- Ciphertext (C): Encrypted form

Cryptographic Keys

- A key is a secret used to perform encryption and decryption
- Decrypting using the key is easy
- Reduces the secrecy problem with long messages and short keys

Cryptosystem Terminology

- Encryption algorithm called E()
- C = E(K, P)
- Decryption algorithm is referred to as D()
- Decryption algorithm also has a key
- Cryptosystem: the combination of the encryption and decryption algorithm

Symmetric Cryptosystems

- C = E(K, P)
- P = D(K, C)
- P = D(K, E(K, P))
- Decrypting the encrypted plaintext results in the plain text

Advantages to Symmetric Cryptosystems

- Encryption and authentication are performed in a single operation
    - Only the reader has the key to decrypt
- Well known and trusted keys perform faster than asymmetric key systems
- No centralized authority is needed

Disadvantages of Symmetric Cryptosystems

- Hard to separate encryption from authentication
    - Complicates some signature uses
- Non repudiation hard without servers

    - Hard to take the key back to deny access
- Scaling for internet use can be difficult
- Key distribution issues

Popular Symmetric Ciphers

- Data Encryption Standard (DES)
    - Old US encryption standard
    - Still somewhat used, but weak
- Advanced Encryption Standard (AES)
    - Current US standard and widely used
- Blowfish and other solutions

Symmetric Ciphers and Brute Force Attacks

- DES has 56 bit keys, which is short for modern brute force attacks
- AES is too long for brute force attacks

Asymmetric Cryptosystems

- Public key cryptography (PK)
- Encryptions and decryptions use different keys

Using Public Key Cryptography

- Keys are created in pairs
- One key is kept secret, and the other key is public
- If you want to send an encrypted message, encrypt using their public key
    - They then decrypt using the private key

Authentication using Public Keys

- Signing a message can be done by encrypting it with the private key
- Only I have the private key, so no one else could create it
- Everyone knows the public key, so people can check the claim directly
- This is better than symmetric cryptography since only the sender could have created the message

Issues with PK Key Distribution

- Security of public key cryptography depends on using the right public key
- Need high assurance that a key belongs to a particular person
    - Key distribution infrastructure or certificates are problematic

PK Algorithms

- Based on a math problem for factoring extremely large numbers
- Security less dependent on brute force and more on the complexity of the underlying problem
- Also implies choosing key pairs is complex and expensive

Example Public Key Cipher

- RSA
  - Popular public key library
- Elliptic curve cryptography
  - Better performance

Security of PK Systems

- Based on solving the underlying problem
- Longer the key, more expensive to encrypt and decrypt

Combined Symmetric and Asymmetric Cryptography

- Common to use both in a single session
- Asymmetric cryptography essentially used to bootstrap symmetric crypto
- Use RSA to authenticate and establish a session key
- Use DES or AES with session key for the rest of the transmission
  - Creating a session creates a new session key

Creating secure sessions

- Uses both symmetric and asymmetric cryptography

1. Both party A and B have their own public and private keys, and they have each others public keys
2. A new connection is made and party A makes a session key
    a. A will the encrypt the session key using B’s public key
    b. Encrypted a second time with A’s private key
    c. This is sent to B
3. B decrypts using A’s public key and B’s private key