Protection and Security

Alexandru Andrei
anel@ida.liu.se
phone: 013-322828, room: B 3D:439

Lecture Plan

1. What is an operating system? What are its functions?
   Basics of computer architectures. (Part I of the textbook)
2. Processes, threads, schedulers (Part II, chap. III-V)
3. Synchronization & Deadlock (Part II, chap. VI, VII)
4. Primary memory management. (Part III, chap. VIII, IX)
5. File systems and secondary memory management (Part IV, chap. X, XI, XII)

Outline

- Protection
  - Protection Domains
  - Access matrix, access control lists, capability lists
- Security
  - Types of attacks
  - Password protection
  - Cryptography
  - Trojans, exploits, worms, viruses

Protection

- Processes must be protected from each other’s activities
- Protection refers to a set of mechanisms used to ensure that resources (CPU, memory, I/O, files) are accessed only after proper authorization by the OS
- OS has to provide means to
  - Specify access control
  - Enforce them

Policies and Mechanisms

- Policies
  - Specify what will be done
  - May change over time
- Mechanisms
  - Specify how policies will be implemented
  - Preferably general, such that a change in policy does not imply a change in the mechanisms (i.e. a change in the OS)
  - Example:
    - Having a firewall represents a mechanism
    - The rules in the firewall represent the policy
OS consists of a collection of
- Processes
- Objects (hw: CPU, memory, I/O; sw: files, semaphores, programs)
- A protection domain specifies
  - The objects that may be used by a process
  - The operations that may be invoked on each object

Kernel mode = monitor mode in the book
In kernel mode the process may execute privileged
instructions (read/write to I/O ports, changing protection
domains, process priorities, etc.)

Why should low level I/O port operations protected?

Protection Domains

Kernel mode = monitor mode in the book
In kernel mode the process may execute privileged
instructions (read/write to I/O ports, changing protection
domains, process priorities, etc.)

Why should low level I/O port operations protected?

Protection Domains

Domains can be associated per
- User
- Process
- Procedure
- Association between processes and domains can be
dynamic
- E.g.: `seteuid(new_uid)` (set effective user ID, is a
  privileged operation, i.e. only root may successfully
  execute it)
- "Need to know basis"

Protection Domains

Domains can be associated per
- User
- Process
- Procedure
- Association between processes and domains can be
dynamic
- E.g.: `seteuid(new_uid)` (set effective user ID, is a
  privileged operation, i.e. only root may successfully
  execute it)
- "Need to know basis"

The `setuid` Bit

Even though `/bin/ls` is owned by root, when a user
executes it, it runs with the user’s ID, thus it has the rights
of the user executing it, not of the owner of the file

If it were:
```
-r-xr-xr-x 1 root  22584 Jan 23 2005 /bin/ls
```
Then it would run with the effective user ID of the file
owner, i.e. root

Examples? Why would we want that?

The `setuid` Bit

Even though `/bin/ls` is owned by root, when a user
executes it, it runs with the user’s ID, thus it has the rights
of the user executing it, not of the owner of the file

If it were:
```
-r-xr-xr-x 1 root  22584 Jan 23 2005 /bin/ls
```
Then it would run with the effective user ID of the file
owner, i.e. root

Examples? Why would we want that?
Access Matrix

<table>
<thead>
<tr>
<th></th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>Laser printer</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>read</td>
<td>read</td>
<td>switch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td>print</td>
<td>switch</td>
<td>switch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3</td>
<td>read</td>
<td>exec</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D4</td>
<td>read</td>
<td>write</td>
<td>switch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Access Control Lists and Capability Lists

- An access control list is a column in the access matrix, i.e. for each object it specifies <process, rights>
  - Attached to an object
- A capability list is a row in the access matrix, i.e. for each domain it specifies <object, rights>
  - Attached to a domain

Who Ensures Protection?

- OS (via various mechanisms: for example firewalls, file permissions, process permissions)
- Compilers (does not compile code that contains I/O operations for example)
- Language (Java for example)

Security

- Can be regarded as protection from external threats, from the environment
- Resources to be protected from unauthorized access, malicious destruction, accidental introduction of inconsistencies

Types of Attacks

- Theft of information
- Unauthorized modification of data
- Unlawful use (spam, launching malicious attacks)
- Preventing legitimate use (denial of service)

Breaking In

- Physically (break into computer room, destroy the computers, install sniffing devices, etc.)
- Human (log in as someone else, root preferably)
- Network (intercept/modify data circulating on the net, send data as coming from a legitimate source, launch data flood for (distributed) denial of service attacks)
- OS (exploit bugs in server software such that they will execute a given malicious code)
Prevention

- Physical attacks: lock the doors 😊
- Human
  - Choose difficult to guess passwords
  - Protect the passwords

Unix Password Protection

$cat /etc/passwd
... aleand6649MpkpY0. :100:10:Alexandru:/home/alean:/bin/bash ...

Password Protection

$telnet remote
User: alean
Password: ev@hvtB
ev@hvtB
crypt() /etc/passwd: d6649MpkpY0.
d6649MpkpY0. compare

-crypt gives a one-way encryption, i.e. it is very difficult to obtain the original from the encrypted form

Cryptography

- Never use telnet! (Always ssh!)
- One-way encryption would not help
- Symmetric keys and asymmetric keys cryptography systems
- Symmetric:
  - Same key used for both encryption and decryption
  - How do we transport the key to the other end over an untrustworthy channel?
- Asymmetric:
  - Different keys

Sniffing

What happens in the case of telnet?
Public/Private Key Systems

- Each user has two keys, a public and a private one
- It is very difficult to calculate the private key when only the corresponding public key is known (very difficult = years of trying on supercomputers)
- Public/Private key systems are computationally more complex than symmetric key approaches

Public/Private Keys

- What is encrypted with the public key can be decrypted only by the private key
- Alice tells everybody the public key. Bob writes her a message and encrypts it with Alice’s public key. Only Alice may read the message because only she has the private key

RSA

- Choose two very large prime numbers \( p \) and \( q \) (512 bits)\n- \( N=p\cdot q \)
- Select the public \( K_e \) key, such that \( K_e \) is relatively prime to \( (p-1)\cdot(q-1) \)
- Select \( K_d \) such that \( K_d \cdot K_e \mod ((p-1)\cdot(q-1)) = 1 \)
- Encryption using the public key: \( E(m) = m^{K_e} \mod N \)
- Decryption using the private key: \( m = E^{K_d} \mod N \)
- It is very difficult to find out the values of \( p \) and \( q \), given the value of \( N \)

RSA: Example

- Choose \( p=7 \) and \( q=13 \)
- \( N=p\cdot q=91; (p-1)\cdot(q-1)=6\cdot12=72 \)
- Select the public \( K_e = 5 \) (5<72 and \( (5,72)=1 \))
- Select \( K_d = 29 \)
- Encrypt message \( m=69 \): \( E(69)=69^5 \mod 91 = 62 \)
- Decryption using the private key: \( m=62^{29} \mod 91 = 69 \)

Digital Signatures

- What is encrypted with private key can be decrypted only by the public key
- Alice computes a hash of a message (the so-called message digest), then she encrypts the digest with her private key, and appends the encrypted digest to the plain text message
- Everybody having Alice’s public key may decrypt the message digest, and compute it from the original plain-text message. If the decrypted and computed digests are identical, they are sure that
  - Alice wrote the message
  - Nobody changed the message on the way

Sniffing

What happens in the case of telnet?
**Ssh with Password Authentication**

- Server sends its public key
- User sends a symmetric session key encrypted with the server’s public key. Only the real server may decrypt it.
- From then on, the symmetric key is used for communication
- The password does not traverse the network in plain text!

**Impersonating**

Does `ssh` with password authentication protect in this case?

Passwords may be read by a corrupted `ssh` server

**Spoofing**

Can a malicious user read the password?
Yes, if the client has no previous knowledge of the real public key of the server

**Ssh with Public/Private Key Authentication**

- A secure channel (encrypted by a symmetric key) is established like in the password authentication scheme
- Next, the server sends a challenge encrypted with the user’s public key
- The legitimate user is the only one to be able to decrypt the challenge
- The response to the challenge is sent back, and if they match, the user is authenticated

**Impersonating**

Even if the server is corrupted, no passwords are sent

**Spoofing**

Can users be duped to think that they connect to the real server?
Yes, if the client has no previous knowledge of the real public key of the server, but no password is sniffed
Certificates

- How can users know that the public key they get from a server is the public key of the real server?
- By means of certificates
  - An authority, the CA (certificate authority), signs the public key of the real server with the private key of the CA
  - Everybody may decrypt the signature and read the public key of the real server
- But nobody may forge the public key of the real server because they do not have access to the CA’s private key in order to issue a new certificate for the forged key

Trojan Horses

- Use of a program that is believed to do something, but in addition it does something else
- E.g.: Emulate a login screen, an ssh server, etc., log typed keys, read system files and email them, etc.

Protection:
- Do not execute untrusted code
- Do not put the current directory (the dot '.') in your search path

Exploits

- Stack and buffer overflows
- E.g.:
  - Safe: char s[BUF_MAX]; fgets(s, BUF_MAX, stdin);
  - Unsafe: char s[BUF_MAX]; gets(s);
- What if one introduces a string that is longer than BUF_MAX? Those characters will overwrite code. What if the extra characters typed are themselves code?

Program Threats

- Trojan horses
- Trap doors
- Exploits

Trap Doors

- Doors left open by the programmer
- E.g.:
  
  ```
  if (uid == alean)
      grant_access();
  else
      authenticate();
  ```

Protection:
- Easy to spot if we have access to the source code of system programs and/or OS
- Difficult to spot if they are very clever, for example the trap is not in the source of the program but in the source of the compiler that compiles the program!

Non-executable Stack

```c
#include <stdio.h>
#include <string.h>
typedef void (*fptr)(void);
#ifdef __sparc
char shellcode[] =
  "\x2d\x0b\xd8\x9a\xac\x15\xa1\x6e\x2f\x0b\xdc\xda\x90\x0b\x80\x0e\x92\x03\xa0\x08\x94\x1a\x80\x0a\x9c\x03\xa0\x10\xec\x3b\xbf\xf0\xdc\x23\xbf\xf8\xc0\x23\xbf\xfc\x82\x10\x20\x3b\x91\xd0\x20\x08";
#endif
int main(int argc, char **argv) {
  fptr f;
  char code[100];
  memcpy(code, shellcode, sizeof(shellcode));
  printf("Attempting to start a shell...\n");
  f = (fptr)code;
  f();
  return (0);
}
```
Non-executable Stack

$ cc -o shell-exstk shell.c
$ cc -o shell-noexstk -M /usr/lib/ld/map.noexec shell.c
$ ./shell-exstk
Attempting to start a shell...
$ exit
$ ./shell-noexstk
Attempting to start a shell...
Segmentation Fault(coredump)
Sep 16 15:06:06 kilroy genunix: [ID 533030 kern.notice]
NOTICE: shell-noexstk[23132] attempt to execute code on stack by uid 101

Stacks can be globally configured to be non-executable using the noexec_user_stack tunable in /etc/system.

A. Andrei, Process programming and operating systems, Security

System Threats

- Viruses
  - Code appended to legitimate programs
- Worms
  - Stand-alone programs

Denial of Service

- For example, starting millions of web browsers that connect simultaneously to the same address
- Do not gain information
- Disable legitimate users
- Very difficult if not impossible to counter

Summary

- Protection
  - Protection Domains
  - Access matrix, access control lists, capability lists
- Security
  - Types of attacks
  - Password protection
  - Cryptography
  - Trojans, exploits, worms, viruses

A. Andrei, Process programming and operating systems, Security
Reading Material

- Silberschatz & Galvin & Gagne, part V
  - Chapter 14: 14.1-14.4, 14.6, 14.9,
  - Chapter 15: 15.1-15.5

Exam

- Written examination
- Read the slides and the recommended chapters from the book
- Use English or Swedish

Good luck!