Agentúra Ministerstva školstva, vedy, výskumu a športu SR pre štrukturálne fondy EÚ

Operačný program VZDELÁVANIE

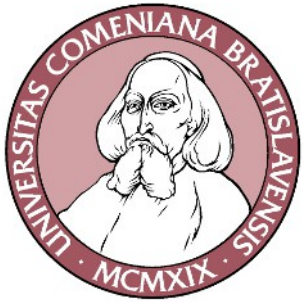
Európska únia
Európsky sociálny fond

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

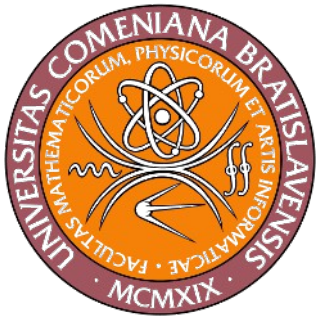
Príprava štúdia matematiky a informatiky na FMFI UK v anglickom jazyku

ITMS: 26140230008

dopytovo – orientovaný projekt



Moderné vzdelávanie pre vedomostnú spoločnosť/Projekt je spolufinancovaný zo zdrojov EÚ



Security of IT infrastructure

Access Control in Operating Systems

RNDr. Jaroslav Janáček, PhD.

Discretionary Access Control (DAC)

- has been a standard feature in many common OS's for long time
- an object's owner specifies access rights for other subjects
- every process runs on behalf of a user
  - and has all rights of the user
  - including the right to specify access rights

Discretionary Access Control

- UNIX/Linux
  - rights (permissions): read, write, execute / use
  - subjects: user, group, others
    - in classic UNIX systems only the owner and 1 group
    - ACL – adds the possibility to specify rights for arbitrary number of groups and users, and to specify the default rights for new objects in a directory

Discretionary Access Control

- Windows
  - finer-grained access rights
  - subjects: users, groups
  - inheritance of rights from higher levels of the directory hierarchy
    - allow/deny rights
      - hierarchically closer rights have precedence
      - deny has precednce over allow if specified on the same level of hierarchy
  - rights of all user's groups are added together

Windows DAC

	full control	modify	read&execute	list folder	read	write
traverse/execute	x	x	x	x		
list folder/read data	x	x	x	x	x	
read attributes	x	x	x	x	x	
read extended attr.	x	x	x	x	x	
create files/write data	x	x				x
create folders/append	x	x				x
write attributes	x	x				x
write extended attr.	x	x				x
delete subfolders&files	x					
delete	x	x				
read permissions	x	x	x	x	x	x
change permissions	x					
take ownership	x					

Insufficiency of DAC

- a user runs a vulnerable application and processes a malicious document
  - the application begins to execute malicious code with the user's rights
  - it has access to all data on behalf of the user
- a user (un)intentionally specifies incorrect access rights
  - other users gain access to data
  - important issue in the world of classified information

User's Rights Misuse

- most serious in the case of users with high access rights
  - UNIX/Linux root
  - Windows Administrators
- natural protection
  - minimize the set of processes with such rights
  - even the minimal set can still be too large

Minimizing the Process's Rights

- Windows Vista / 7
  - UAC (User Account Control)
    - an attempt to use an administrator's rights leads to a request for explicit approval by the user
- Linux
  - capabilities
    - a way of splitting the root's privileges
    - most privileged processes only need a small subset of the root's privileges
    - fully supported since 2.6.24 Linux kernel
      - not often used, however

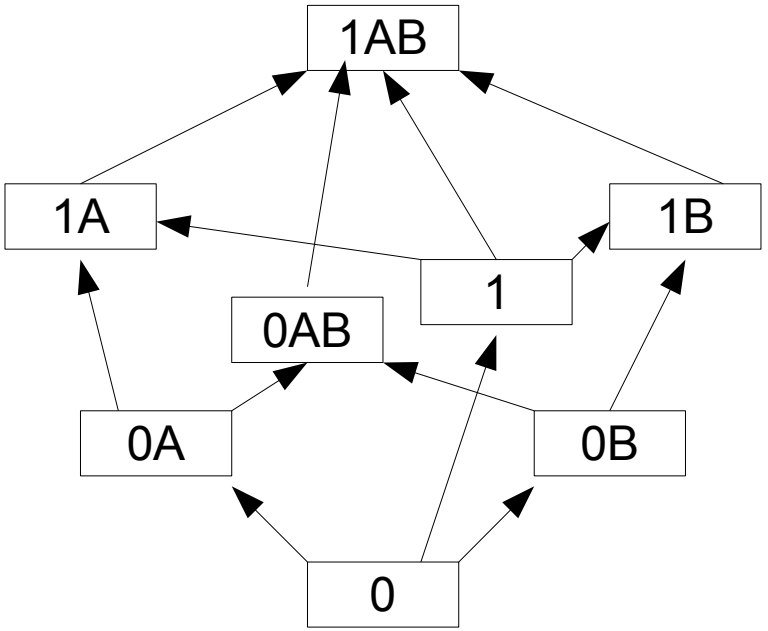
Mandatory Access Control (MAC)

- basic idea
  - access restricted by a policy that normal processes and users cannot change
  - malicious code executed in a process's context can only perform operations that the policy allows the process to perform
- possible use
  - processes can only perform limited operations
    - and have limited impact on the system (even when some vulnerabilities are exploited)

Bell – La Padula Model

- from the classified information world
  - confidentiality protection
  - pieces of information are labeled
    - sensitivity level *s* (a value from an ordered set)
    - set of categories *C*
  - subjects have a clearance
    - max. sensitivity level, set of categories
  - labels are partially ordered
    - $(s_1, C_1) \geq (s_2, C_2) \Leftrightarrow s_1 \geq s_2 \ \wedge C_1 \supseteq C_2$
    - not all labels are comparable

Bell – La Padula Model



Bell – La Padula Model

- given an object with a sensitivity level O and a subject with a sensitivity level S, the subject can
  - read from the object, if $S \geq O$
    - no read up
  - modify the object, if $O \geq S$
    - no write down
    - in some systems only if $O = S$
- special – trusted subjects are not restricted by the second condition
  - they can decrease the label of the information

Bell – La Padula Model

- a subject can have a range of lables
  - current
    - used for the access control
  - maximal
    - the subject can increase its current label upto this one
    - but it cannot decrease its current label

Biba Model

- integrity protection
  - instead of sensitivity level we have a trustworthiness level
  - inverse rules to those of Bell – La Padula
    - no read down, no write up
  - it ensures that
    - subjects with a lower label cannot modify data with a higher label
    - subjects with a higher label cannot be influenced by data with a lower label

Windows Vista/7 MIC

- Mandatory Integrity Control
  - implements a **part** of Biba model
  - levels Low, Medium, High, System
  - only no write up
  - optionally no read up (Bell – La Padula)
  - intended to provide protection against malicious modification of data by code from untrustworthy sources (e.g. from the Internet)

Domain and Type Enforcement

- subjects are assigned a **domain**
- objects are assigned a **type**
- the policy specifies
  - operations that a subject within a domain can perform on an object of a given type
  - allowed transitions between domains
  - the type of a new object based on the domains of the creating subject and the type of the parent object

SELinux

- DTE
  - makes no formal distinction between domains and types
- Bell – La Padula (or Biba)
  - the rules are configurable
- Role Based Access Control
  - a role has a set of allowed domains
  - a user has a set of allowed roles
    - UNIX and SELinux user identities are distinct, SELinux identity is assigned based on a mapping

SELinux

- every subject (process) is assigned a context
  - user:role:type[:mls_range]
- policy
  - types, attributes
    - attributes are used to label a set of types
  - rules to determine the type for a new object/subject
  - allowed operations
  - roles, allowed domains
  - users, allowed roles
  - constraints

SELinux

- contexts for filesystem objects
  - stored in extended attributes
  - requires support from the filesystem
- contexts for some classes of objects
  - defined using special rules in the policy (e.g. for network ports)
  - derived from the contect of the process that creates the object

SELinux

```
attribute domain;
attribute files_type;
type user_t, domain;
type spec_t, domain;
type spec_exec_t;
type_transition domain spec_exec_t:process spec_t;
allow domain spec_t:process transition;
allow spec_t spec_exec_t:file entrypoint;
allow domain spec_exec_t:file {read execute};
allow spec_t files_type:file *;

role user_r types {user_t spec_t};
user user_u roles user_r;
```

SELinux

- reference policy
  - targeted vs. strict
    - nowadays common – combined
    - unconfined_u:unconfined_r:unconfined_t
      - unrestricted
  - modular – a module
    - defines its types, attributed, rules
    - defines interfaces that can be used by other modules
    - defines contexts for files

SELinux

- chcon
  - change an object's context
- runcon
  - execute a program in the given context
- semodule
  - load/remove policy modules
- semanage
  - administration of some parameters
- /etc/selinux/...

AppArmor

- profiles for restricted applications
  - described in a simple text form
  - define the access rights to files
  - can define transitions to other profiles on execution of another program
- does not use extended attributes in the filesystem
- the entire configuration is stored in the profiles
  - a binary form is loaded into the kernel

AppArmor

```
/usr/bin/prog {
  /path/to/file  rw,
  /etc/**        r,
  /bin/*         ix,
  /usr/bin/prog1 px,
  network inet stream
 }
```

Demo

- capabilities
  - setcap cap_net_raw=pe ping
    - allows the process to use raw sockets
- SELinux MLS
  - chcon -l s0:c0 x.txt
  - runcon -l s0 bash
    - we will not have access to x.txt
      - s0:<empty cats> is not >= s0:c0

Demo

- SELinux
  - semanage user ...
    - management of SELinux users and their allowed roles
  - semanage login ...
    - management of the mapping between UNIX and SELinux identities
  - semanage login -a -s user_u user
    - the user *user* will be assigned SELinux identity of *user_u*
  - newrole -r sysadm_r
    - change the current role