

Mecanismos de controle de acesso

Segurança Computacional

Prof. Carlos Maziero

DInf UFPR, Curitiba PR

Setembro de 2026

Conteúdo

- 1 Infraestrutura de controle de acesso
- 2 Controle de acesso em UNIX
- 3 Controle de acesso em Windows
- 4 Outros mecanismos de controle de acesso
- 5 Mudança de privilégios

Infraestrutura de controle de acesso

Infraestrutura de controle de acesso

Um mecanismo de controle de acesso deve ser:

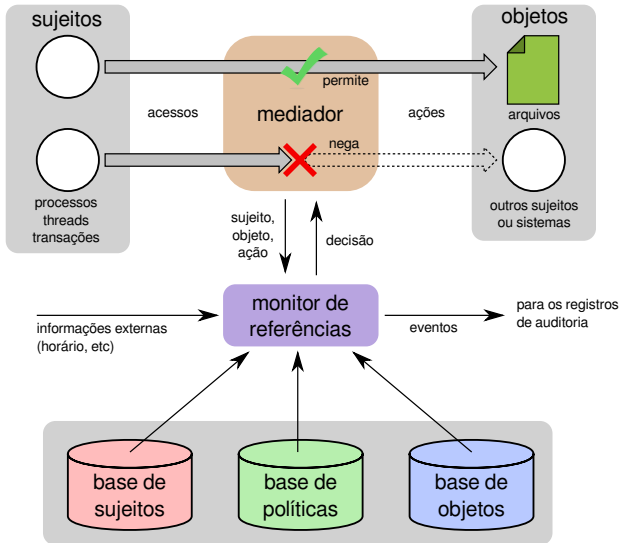
- independente da política de controle de acesso
- inviolável: impossível de adulterar ou enganar
- incontornável: deve mediar todos os acessos

Infraestrutura de controle de acesso

Elementos da infra-estrutura de controle de acesso:

- Base de sujeitos e objetos, com seus respectivos atributos
- Base de políticas de controle de acesso
- Monitor de referências: julga cada pedido de acesso
- Mediador (*Enforcer*): medeia os pedidos de acesso

Infraestrutura de controle de acesso



Controle de acesso em UNIX

Controle de acesso em UNIX

Mecanismo tradicional: ACLs

- Três sujeitos: *user*, *group*, *others*
- Três possibilidades de acesso: *read*, *write*, *execute*
- Apenas 9 bits para definir as permissões básicas
- Aplicado somente na “abertura” do recurso

```

1 host:~> ls -l
2 d rwx --- --- 2 maziero prof      4096 2008-09-27 08:43 figuras
3 - rwx r-x --- 1 maziero prof      7248 2008-08-23 09:54 hello-unix
4 - rw- r-- r-- 1 maziero prof         54 2008-08-23 09:54 hello-unix.c
5 - rw- --- --- 1 maziero prof         59 2008-08-23 09:49 hello-windows.c
6 - rw- r-- r-- 1 maziero prof    195780 2008-09-26 22:08 main.pdf
7 - rw- --- --- 1 maziero prof    40494 2008-09-27 08:44 main.tex
  
```

Controle de acesso em UNIX

	tipo (arquivo, diretório, atalho, ...)	permissões (proprietário)	permissões (grupo)	permissões (terceiros)	número de ligações	proprietário	grupo						
d	rw-	---	---	2	maziero	prof	4096	2008-09-27	08:43	figuras			
-	rw-	r-x	---	1	maziero	prof	7248	2008-08-23	09:54	hello-unix			
-	rw-	r--	r--	1	maziero	prof	54	2008-08-23	09:54	hello-unix.c			
					tamanho em bytes			data/hora da última modificação					nome

ACLs estendidas

Definidas no padrão POSIX 1003.1e

Permitem granularidade mais fina de controle de acesso

Adotadas na maioria dos sistemas UNIX (Linux, MacOS, etc.)

```
1 host:~> ll
2 -rw-r--r-- 1 maziero prof 24791 Jul 26 10:47 main.pdf
3
4 host:~> getfacl main.pdf
5 # file: main.pdf
6 # owner: maziero
7 # group: maziero
8 user::rw-
9 group::r--
10 other::r--
```

ACLs estendidas

```
1 host:~> setfacl -m diogo:rw,rafael:rw main.pdf
```

```
2  
3 host:~> getfacl main.pdf
```

```
4 # file: main.pdf
```

```
5 # owner: maziero
```

```
6 # group: maziero
```

```
7 user::rw-
```

```
8 user:diogo:rw-
```

```
9 user:rafael:rw-
```

```
10 group::r--
```

```
11 mask::rw-
```

```
12 other::r--
```

Controle de acesso em Windows

Controle de acesso em Windows

Sujeitos:

- Sujeitos: computador, usuário, grupo ou domínio
- SID - *Security IDentifier*
- Cada sujeito recebe um *access token*, criado no login

Access Token (AT):

- Atribuído a um processo e herdado por seus filhos
- contém o SID do usuário e dos grupos aos quais pertence
- privilégios associados a ele (*reboot, debug, etc*)
- outras informações

Controle de acesso em Windows

Objetos:

- Arquivos, processos, serviços, chaves de registros
- Cada objeto está associado a um *Security Descriptor*

Security descriptor:

- Proprietário e o grupo primário do objeto
- DACL - *Discretionary ACL*
- SACL - *System ACL*
- Informações de controle diversas

DACL

DACL: *Discretionary ACL*

- Lista de regras de controle de acesso (ACEs)
- ACE: *Access Control Entry*:
 - Identificador de usuário ou grupo
 - Modo de autorização (positiva ou negativa)
 - Conjunto de permissões (ler, escrever, executar, etc)

SACL

SACL: *System ACL*

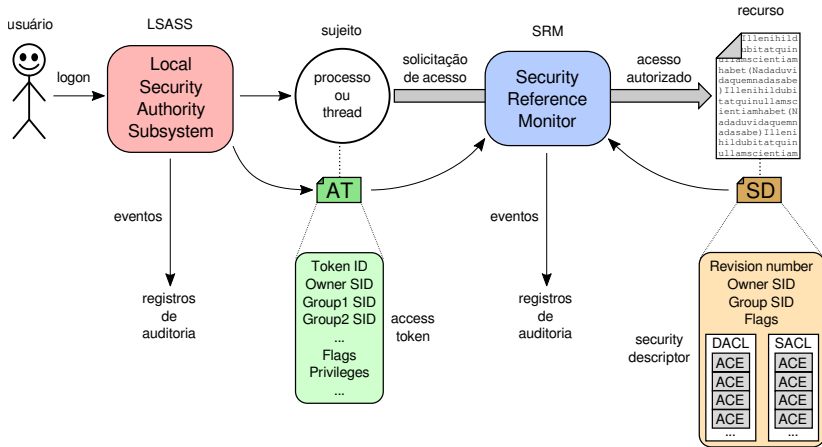
- Usada para fins de auditoria
- Define operações devem ser registradas nos logs
- Similar à estrutura das ACEs da DACL

Controle de acesso em Windows

Principais entidades:

- LSASS - *Local Security Authority Subsystem*:
 - Autenticação e início da sessão do usuário
 - Criação dos processos iniciais
 - Associa os processos a *access tokens* (AT)
- SRM - *Security Reference Monitor*:
 - Intermediação dos acessos aos recursos
 - Verificação das ACLs dos objetos
 - Compara o AT do sujeito com a DACL do objeto
 - Pode acionar mecanismo de resolução de conflitos

Controle de acesso em Windows



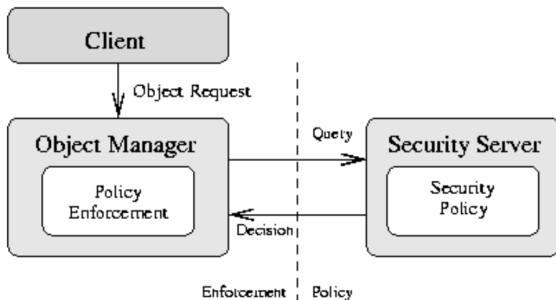
Outros mecanismos de controle de acesso

SELinux

- SELinux: *Security Enhanced Linux*
- Desenvolvido pela *National Security Agency* – USA
- Capaz de implementar políticas MLS e MCS
- Política default baseada em RBAC e DTE
- Complexo de gerenciar...

SELinux

Inspirado na arquitetura de segurança *Flask*



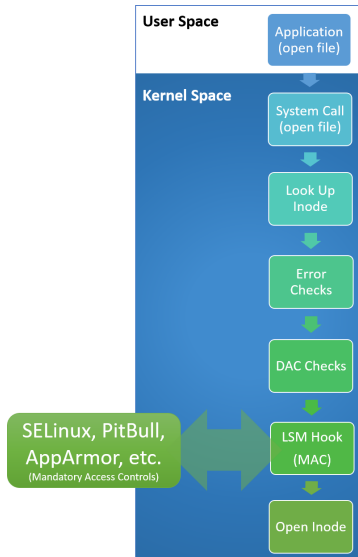
Fonte: *The Flask Security Architecture: System Support for Diverse Security Policies*. R. Spencer et al, USENIX Security Symposium, 1999.

SELinux

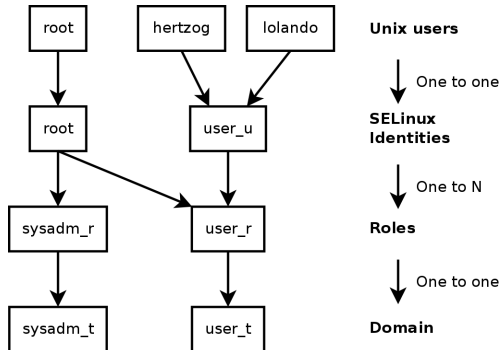
Usado em sistemas Linux e Android

Usa os “ganchos” LSM
(*Linux Security Modules*)

Fonte: *PitBull and SELinux Mandatory Access Control Systems*. F. Caviggia, The MITRE Corporation, 2018.



SELinux



Fonte: *The Debian Administrator's Handbook*. R. Hertzog et al, Debian Project, 2015.

Windows MIC

MIC: Mandatory Integrity Control

Implementa o modelo MAC de Biba

Associa aos processos e recursos níveis de integridade:

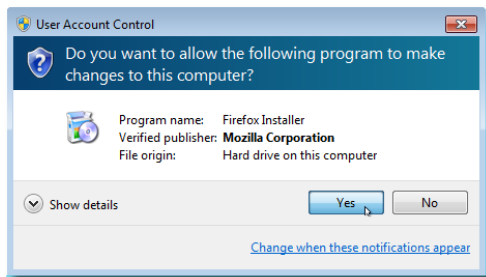
- *Low*: navegador Web e executáveis provindos da Internet
- *Medium*: processos normais dos usuários
- *High*: processos/usuários administrativos
- *System*: serviços do sistema

Windows UAC

UAC: *User Account Control*

Aplica uma política baseada em RBAC

Papéis administrativos só são ativados quando necessários



Trusted BSD

Implementa ACLs no padrão POSIX e capacidades POSIX

Suporta MAC: Bell LaPadula, Biba, categorias, DTE

Portado para o MacOS X (*MacOS X MAC Framework*)

Trusted Solaris

Multi-Level Security: níveis de segurança são associados aos recursos e usuários

Usa também domínios, implementados através de “compartimentos”

Um recurso associado a um determinado compartimento só pode ser acessado por sujeitos no mesmo compartimento (DTE)

Usa RBAC para organizar papéis de administração do sistema

Outros mecanismos

Mecanismos locais:

- AppArmor
- Capsicum
- Smack
- LOMAC
- PAX
- GRSecurity
- PolicyKit
- ...

Mecanismos distribuidos:

- OAuth
- XACML
- NGAC
- SAML
- PERMIS
- Apache Shiro
- Google Zanzibar
- ...

Mudança de privilégios

Mudança de privilégios

Processos são sujeitos que representam usuários:

- Cada processo herda as **credenciais** de seu pai
- Cada processo herda as **permissões** de seu pai
- Adequado para o uso normal do sistema

Entretanto, esse modelo não funciona caso o usuário precise:

- Instalar um novo programa
- Atualizar sua senha

Essas ações necessitam privilégios mais elevados!

Abordagens para mudança de privilégios

Usuários administrativos:

- Permissões administrativas para certos usuários
- Privilégios administrativos podem ser divididos
- Problemático se conta do usuário for comprometida

Exemplo: Windows XP

Abordagens para mudança de privilégios

Permissões (ou papéis) temporárias:

- Conceder permissão para realizar ação administrativa
- Permissão descartada pelo processo ao concluir a ação
- Permissões são associadas a papéis administrativos
- Ativação do papel pode impor reautenticação

Exemplo: infra-estrutura Windows UAC (*User Access Control*)

Abordagens para mudança de privilégios

Mudança de credenciais:

- Permitir que um processo mude de identidade
- É uma variante da cessão de permissões temporárias

Exemplo: flags `setuid` e `setgid` do UNIX

Abordagens para mudança de privilégios

Monitores:

- Processos privilegiados responsáveis pelas ações
- Recebem pedidos de processos não-privilegiados
- Pedidos dos processos são validados e atendidos
- Respeita o princípio da *separação de privilégios*

Exemplo: infra-estrutura *PolKit* do Desktop Linux

Flags SetUID e SetGID

Mecanismo para mudança de credenciais em sistemas UNIX

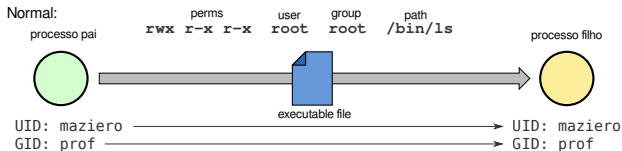
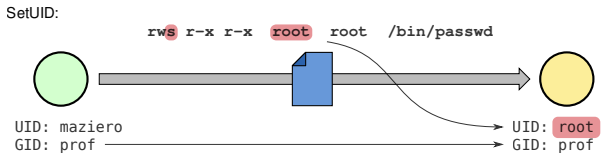
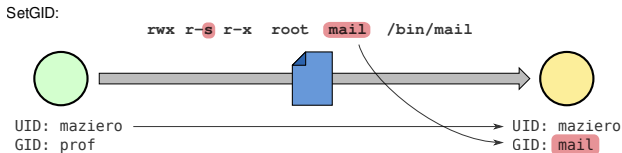
Aplicado às permissões de arquivos **executáveis**:

- Flag `setuid`: o processo herda a credencial do **proprietário** do arquivo executável
- Flag `setgid`: o processo herda a credencial do **grupo**

Os flags são ajustados usando o comando `chmod`:

- `chmod u+s programa.exe`
- `chmod g+s programa.exe`

Flags setuid e setgid



Flags setuid e setgid

Muito usados em programas administrativos no UNIX (troca de senha, agendamento de tarefas, etc)

Vantagens:

- Simplicidade do conceito
- Implementação eficiente

Problemas:

- O processo filho recebe **todos** os privilégios!
→ **violação do princípio do privilégio mínimo**
- Um executável com bugs pode comprometer o sistema

Flags setuid e setgid

Vários programas executam com esses flags ativos:

```

1  -rwsr-xr-- 1 root dip 428424 fev 17 2025 /usr/sbin/pppd
2  -rwsr-xr-x 1 root root 39224 jul 16 2025 /usr/bin/fusermount3
3  -rwsr-xr-x 1 root root 70888 abr 19 2025 /usr/bin/chfn
4  -rwsr-xr-x 1 root root 30952 jan 17 2025 /usr/bin/pkexec
5  -rwsr-xr-x 1 root root 52936 abr 19 2025 /usr/bin/chsh
6  -rwsr-xr-x 1 root root 84360 mai 9 2025 /usr/bin/su
7  -rwsr-xr-x 1 root root 118168 abr 19 2025 /usr/bin/passwd
8  -rwsr-xr-x 1 root root 88568 abr 19 2025 /usr/bin/gpasswd
9  -rwsr-xr-x 1 root root 166848 abr 16 18:27 /usr/bin/ntfs-3g
10 -rwsr-xr-x 1 root root 72072 mai 9 2025 /usr/bin/mount
11 -rwsr-xr-x 1 root root 55688 mai 9 2025 /usr/bin/umount
12 -rwsr-xr-x 1 root root 306456 fev 11 16:22 /usr/bin/sudo
13 -rwsr-xr-x 1 root root 18816 mai 9 2025 /usr/bin/newgrp
14 ...
  
```

Bugs nesses programas podem ser muito perigosos!

Privilégios POSIX

Do inglês *POSIX Capabilities* (mas **não são** capacidades!)

Alternativa mais segura aos flags `setuid` e `setgid`

Poder do super-usuário é fracionado:

- dividido em pequenos privilégios específicos
- Geralmente associados a ações administrativas
- um processo pode receber apenas alguns privilégios
- Podem ser ativados/desativados pelo processo

Privilégios POSIX

- CAP_CHOWN: alterar o proprietário de qualquer arquivo
- CAP_USER_DEV: abrir dispositivos em `/dev`
- CAP_USER_FIFO: usar *pipes* (comunicação)
- CAP_USER SOCK: abrir *sockets* de rede
- CAP_NET_BIND_SERVICE: abrir portas de rede (< 1024)
- CAP_NET_RAW: abrir *raw sockets*
- CAP_KILL: enviar sinais para processos de outros usuários
- ... (outros +30 privilégios)

Privilégios POSIX

Três conjuntos de privilégios por processo:

- **Permitidos** (P): aqueles que o processo pode ativar
- **Efetivos** (E): aqueles ativados no momento ($E \subseteq P$)
- **Herdáveis** (H): transmitidos aos processos-filhos ($(E \subseteq P)$)

Podem ser atribuídos aos executáveis em disco:

- Substituem os flags `setuid` e `setgid`
- O novo processo recebe um conjunto de privilégios calculado a partir dos privilégios atribuídos ao executável e aqueles herdados do processo-pai

Exemplo: o comando ping

Ping: verifica acessibilidade de *hosts* na rede

```
1  $ ping www.ufpr.br
2  PING web.ufpr.br (200.17.209.3) 56(84) bytes of data.
3  64 bytes from 200.17.209.3: icmp_seq=1 ttl=61 time=0.354 ms
4  64 bytes from 200.17.209.3: icmp_seq=2 ttl=61 time=0.396 ms
5  64 bytes from 200.17.209.3: icmp_seq=3 ttl=61 time=0.659 ms
6  64 bytes from 200.17.209.3: icmp_seq=4 ttl=61 time=0.696 ms
7  64 bytes from 200.17.209.3: icmp_seq=5 ttl=61 time=1.51 ms
8  ^C
9  --- web.ufpr.br ping statistics ---
10 5 packets transmitted, 5 received, 0% packet loss, time 4075ms
11 rtt min/avg/max/mdev = 0.354/0.724/1.517/0.419 ms
```

Exemplo: o comando `ping`

Para funcionar, `ping` precisa abrir *raw socket* (ICMP)

Somente o *root* pode abrir um *raw socket*

Solução: `ping` executa com o bit *SetUID* ativo

```
1 $ ls -l /usr/bin/ping
2 -rwsr-xr-x 1 root root 64424 jun 28 08:05 /usr/bin/ping
```

Problema: violação do princípio do **privilégio mínimo!**

Exemplo: o comando ping

Solução melhor: usar privilégios POSIX

a) desativar bit *SetUID*:

```
1 $ sudo chmod u-s /usr/bin/ping
2
3 $ ping www.ufpr.br
4 ping: socket: Operation not permitted
```

b) ativar privilégio para abrir *raw sockets*:

```
1 $ setcap cap_net_raw=ep /usr/bin/ping
```

(Desnecessário nos sistemas mais recentes)