# Defesas em Sistemas

Jorge Correia

Segurança Computacional
Universidade Federal do Paraná

2024

Saltzer & Schroeder [SOSP '73, CACM '74]

- Economia de mecanismos - KISS
- Design aberto - Segurança por ofuscação
- Separação de privilégio - *Least Privilege*
- Aceitação psicológica - Se for difícil de usar então não vai ser

# Políticas de Segurança

- Confidencialidade
- Integridade
- Disponibilidade

## Defesas comuns em Sistemas

- Sistemas de controle de acesso
- Sistemas de autenticação
- Logging
- Encriptação em nível de sistema de arquivos

# Canais secretos (Covert Channels)

Fluxos de informação que não são tratados pelos mecanismos de segurança

## Canais secretos (Covert Channels)

Fluxos de informação que não são tratados pelos mecanismos de segurança

- Demonstrar que não existem canais secretos é difícil

## Canais secretos (Covert Channels)

Fluxos de informação que não são tratados pelos mecanismos de segurança

- Demonstrar que não existem canais secretos é difícil
- Exemplo: **Canais temporais**
- Exemplo: **Canais físicos**
  - Energia elétrica
  - Temperatura
  - Ondas eletromagnéticas
  - Ondas sonoras

## Covert Channels vs Side Channels

- Covert Channels: Atacante intencionalmente utiliza o meio para vazar dados
- Side Channel: Atacante utiliza sinais criados por usuários legítimos

# Hardening

## Hardening

Adoção de medidas de seguranças adicionais, normalmente reduzindo a superficie de ataques

- Limitar o acesso administrativos com "sudo"

## Hardening

Adoção de medidas de seguranças adicionais, normalmente reduzindo a superficie de ataques

- Limitar o acesso administrativos com "sudo"
- Remoção de softwares/serviços desnecessários

## Hardening

Adoção de medidas de seguranças adicionais, normalmente reduzindo a superficie de ataques

- Limitar o acesso administrativos com "sudo"
- Remoção de softwares/serviços desnecessários
- Restrição de execução em espaços de memória

## Hardening

Adoção de medidas de seguranças adicionais, normalmente reduzindo a superficie de ataques

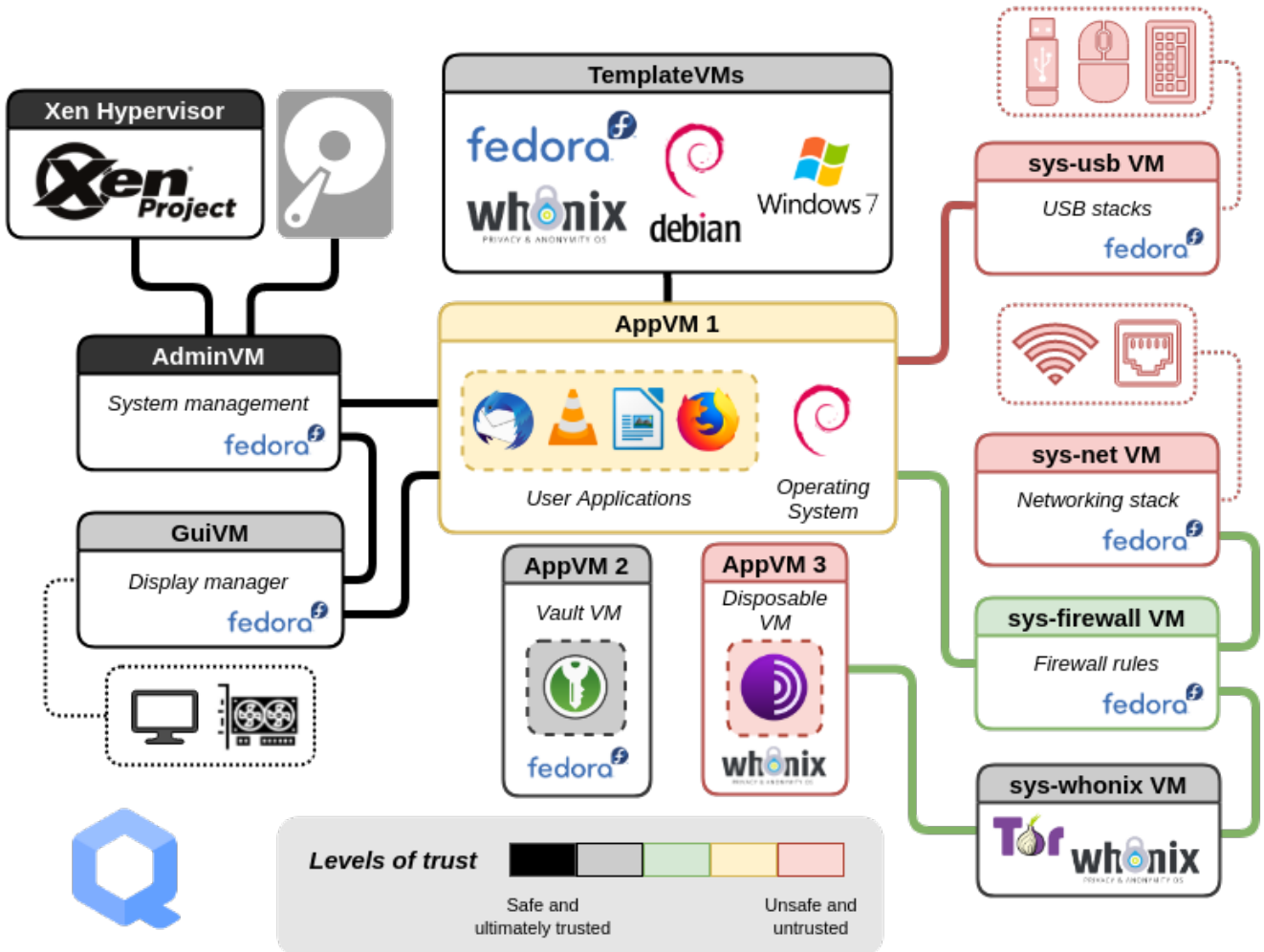
- Limitar o acesso administrativos com "sudo"
- Remoção de softwares/serviços desnecessários
- Restrição de execução em espaços de memória
- Binary hardening
  - Instrumentação de códigos legados
  - Limitação via compilação
  - Limitação via ambiente de execução (eBPF)

## Hardening

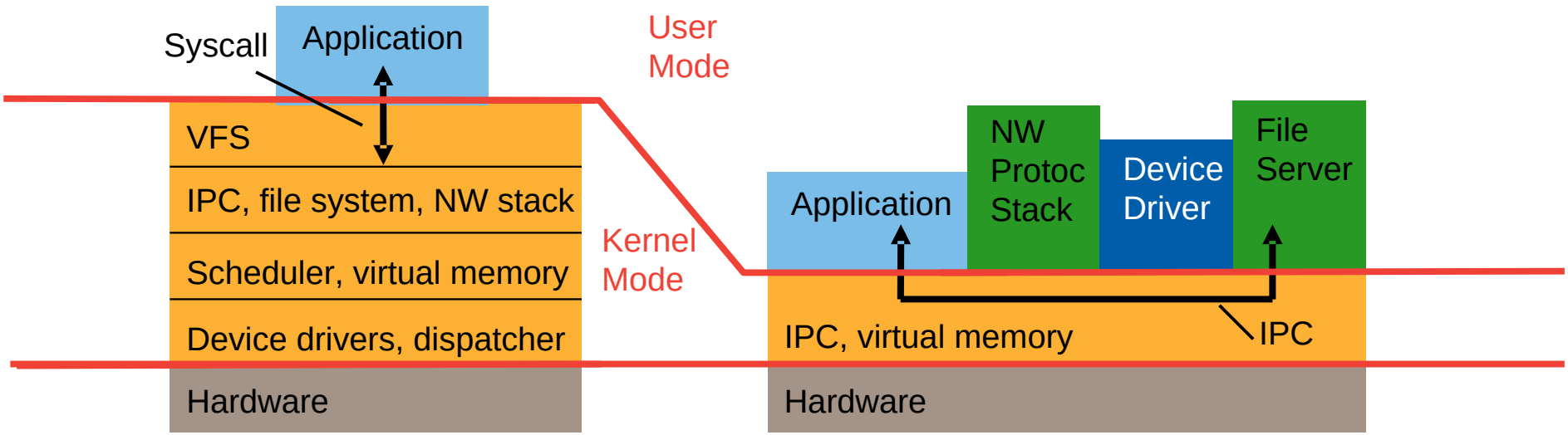
Adoção de medidas de seguranças adicionais, normalmente reduzindo a superficie de ataques

- Limitar o acesso administrativos com "sudo"
- Remoção de softwares/serviços desnecessários
- Restrição de execução em espaços de memória
- Binary hardening
  - Instrumentação de códigos legados
  - Limitação via compilação
  - Limitação via ambiente de execução (eBPF)
- Sandbox
  - Containers
    - Docker
    - **Systemd-nspawn**
  - Máquinas virtuais
    - Xen
    - KVM

## Virtualização



## Microkernel



Hardening em distribuições Linux

- https://wiki.archlinux.org/title/Security
- https://wiki.gentoo.org/wiki/Hardened_Gentoo

# Controle de acesso

Baseado em:
http://www.cse.unsw.edu.au/ cs9242/23/lectures.shtml

**Quem** pode acessar **o que** de que **forma**

- Quem: sujeito, agentes
  - usuários
  - processos
- O que: objetos
  - Arquivos
  - Sockets
  - Processos
- Forma: Permissões
  - Leitura
  - Escrita
  - Execução

- **Políticas** de controle de acesso
  - Especifica acessos permitidos
  - Como esses acessos mudam com o tempo
- **Mecanismos** de controle de acesso
  - Implementam a política

## Matriz de controle de acesso ([Lampson '71])

| | **Obj1** | **Obj2** | **Obj3** | **Subj2** |
|---|---|---|---|---|
| **Subj1** | R | RW | | send |
| **Subj2** | | RX | | control |
| **Subj3** | RW | | RWX<br>own | recv |

## Matriz de controle de acesso ([Lampson '71])

| | **Obj1** | **Obj2** | **Obj3** | **Subj2** |
|---|---|---|---|---|
| **Subj1** | R | RW | | send |
| **Subj2** | | RX | | control |
| **Subj3** | RW | | RWX<br>own | recv |

- Manter essa tabela para todo o sistema é custoso
  - Tamanho
  - Altamente dinâmica

## Controle de acesso por colunas

| | Obj1 | Obj2 | Obj3 | Subj2 |
|---|---|---|---|---|
| **Subj1** | R | RW | | send |
| **Subj2** | | RX | | control |
| **Subj3** | RW | | RWX<br>own | recv |

## Controle de acesso por colunas

| | Obj1 | Obj2 | Obj3 | Subj2 |
|---|---|---|---|---|
| **Subj1** | R | RW | | send |
| **Subj2** | | RX | | control |
| **Subj3** | RW | | RWX<br>own | recv |

Obj1

- Subj1: R
- Subj3: RW

**Access Control Lists (ACL)**

## Controle de acesso por linhas

| | Obj1 | Obj2 | Obj3 | Subj2 |
|---|---|---|---|---|
| **Subj1** | R | RW | | send |
| **Subj2** | | RX | | control |
| **Subj3** | RW | | RWX<br>own | recv |

## Controle de acesso por linhas

| | Obj1 | Obj2 | Obj3 | Subj2 |
|---|---|---|---|---|
| **Subj1** | R | RW | | send |
| **Subj2** | | RX | | control |
| **Subj3** | RW | | RWX<br>own | recv |

Subj3

- Obj1: RW
- Obj3: RWX, own
- Subj3: recv

**Capability (Clist)**

# Access Control Lists (ACLs)

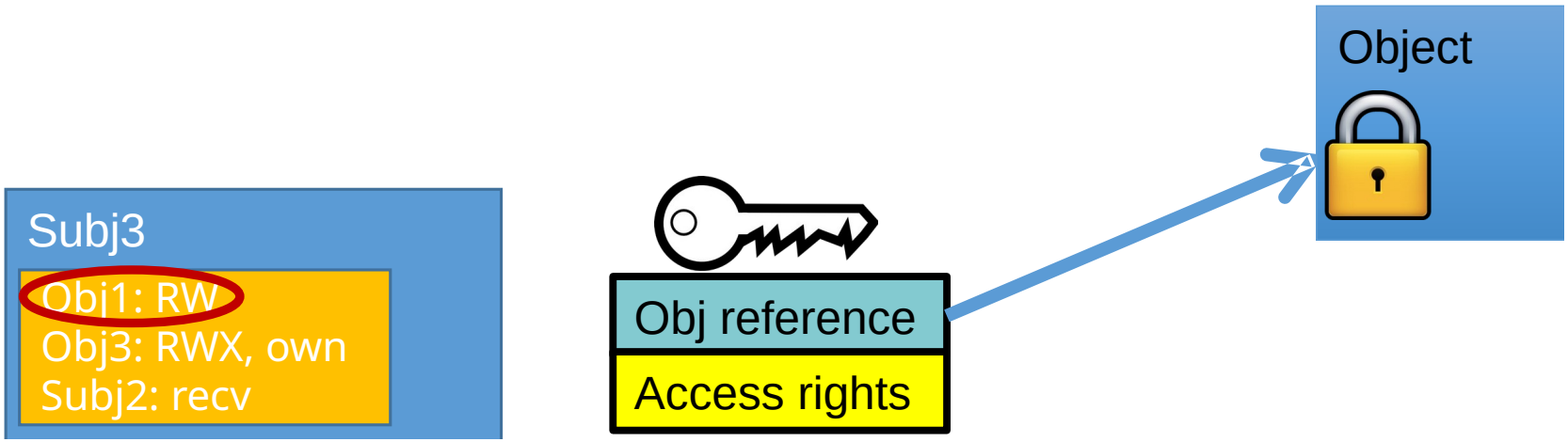
- Sujeitos normalmente são agregados em grupos
  - Owner, Group, everyone
- Podem ter permissões negativas
- Usado na maioria dos SOs mainstream

## Capability-based Access Control

- Token de acesso
- Para acessar um recurso, é preciso apresentar uma capability

# Capability-based Access Control

- Token de acesso
- Para acessar um recurso, é preciso apresentar uma capability



# Capability-based Access Control

- Granularidade fina de controle de acesso
- Fácil delegação de direitos de acesso

## Capability-based Access Control

- Granularidade fina de controle de acesso
- Fácil delegação de direitos de acesso
- Linux recentemente introduziu o conceito de capability, mas não são "object capabilities"

## Capability-based Access Control

- Granularidade fina de controle de acesso
- Fácil delegação de direitos de acesso
- Linux recentemente introduziu o conceito de capability, mas não são "object capabilities"
  - Permitem restringir a quantidade de chamadas de sistemas
  - Podem ter diversos objetos que são acessados por uma chamada de sistema
  - Não existe delegação de direitos de acesso

# ACLs vs Capabilities

- Em teoria, são duas formas de representar a matriz de controle de acesso

## ACLs vs Capabilities

- Em teoria, são duas formas de representar a matriz de controle de acesso
- Diferenças na prática:

## ACLs vs Capabilities

- Em teoria, são duas formas de representar a matriz de controle de acesso
- Diferenças na prática:
  - Name spaces

## ACLs vs Capabilities

- Em teoria, são duas formas de representar a matriz de controle de acesso
- Diferenças na prática:
  - Name spaces
  - setuid

## ACLs vs Capabilities

- Em teoria, são duas formas de representar a matriz de controle de acesso
- Diferenças na prática:
  - Name spaces
  - setuid
  - ACLs dependem de outros mecanismos para compor o sistema de segurança total

## ACLs vs Capabilities

- Em teoria, são duas formas de representar a matriz de controle de acesso
- Diferenças na prática:
  - Name spaces
  - setuid
  - ACLs dependem de outros mecanismos para compor o sistema de segurança total
  - Capability pode ser atribuída ao processo sem relação com o usuário (fork, least privilege, delegação)

## ACLs vs Capabilities

- Em teoria, são duas formas de representar a matriz de controle de acesso
- Diferenças na prática:
  - Name spaces
  - setuid
  - ACLs dependem de outros mecanismos para compor o sistema de segurança total
  - Capability pode ser atribuída ao processo sem relação com o usuário (fork, least privilege, delegação)
- Auditar sistemas

## ACLs vs Capabilities

- Em teoria, são duas formas de representar a matriz de controle de acesso
- Diferenças na prática:
  - Name spaces
  - setuid
  - ACLs dependem de outros mecanismos para compor o sistema de segurança total
  - Capability pode ser atribuída ao processo sem relação com o usuário (fork, least privilege, delegação)
- Auditar sistemas
  - Quem pode acessar um objeto particular

## ACLs vs Capabilities

- Em teoria, são duas formas de representar a matriz de controle de acesso
- Diferenças na prática:
  - Name spaces
  - setuid
  - ACLs dependem de outros mecanismos para compor o sistema de segurança total
  - Capability pode ser atribuída ao processo sem relação com o usuário (fork, least privilege, delegação)
- Auditar sistemas
  - Quem pode acessar um objeto particular
  - Quais objetos um agente pode pode acessar?

# Mandatory (MAC) vs Discretionary Access Control

- Mandatory Access Control (MAC)
  - Administrador impõe as políticas de acesso
  - Aplicações podem ter autonomia dentro da politica imposta pelo administrador
  - Podem prevenir que aplicações maliciosas causem danos
- Discretionary Access Control (DAC)
  - Usuários podem tomar a decisão de controles de acesso
  - Podem delegar seus próprios direitos para outros usuários

# Anti Virus(AV)

# Anti Virus(AV)

- Software utilizado para prevenir, detectar e remover malwares
- Não são perfeitos

# Anti Virus (AV)

- AV products vs AV engines

# Anti Virus (AV)

- AV products vs AV engines
- AV possuem diferentes componentes
  - Kernel Driver
  - Bibliotecas (DLLs)
  - Interface gráfica

# Anti Virus (AV)

- Assinaturas
  - Padrões dos bytes dos arquivos
  - Funcionamento parecido com IDSs
  - Banco de dados de assinatura
  - Fácil bypass. Um malware com novo padrão nunca será detectado
- Heurísticas e Aprendizado de máquina
  - Pode identificar malwares que não possuem assinaturas no banco de dados
  - Falsos positivos (whitelist)

# Anti Virus (AV)

- Verificação estática
  - Realiza a análise sob o binário
- Verificação dinâmica
  - Realiza a análise através de hooks
  - Podem realizar verificações em redes, agindo como um proxy (man-in-the-middle)
- Existem métodos de ofuscação para ambos os métodos
  - Encriptação/encoding de código
  - Máquinas virtuais
  - Modificações de comportamentos
  - Mensagens trocadas pela rede não podem ser desfeitas

Quarentena

- Quando detectado, um malware não é removido, mas sim colocado em quarentena
- Na prática, os AV utilizam uma codificação diferente, então os malwares não são mais executáveis
- Um usuário pode retirá-lo em caso de falso positivo

# Anti Virus (AV)

Vale a pena utilizar AVs?

# Anti Virus (AV)

All You Always Wanted To Know About AntiViruses - Marcus Botacin: https://www.youtube.com/watch?v=fnexx1Ek168