Segurança em Virtualização utilizando o KVM

Klaus Heinrich Kiwi, IBM LTC

Agenda

Introdução

Arquitetura

Gerenciamento

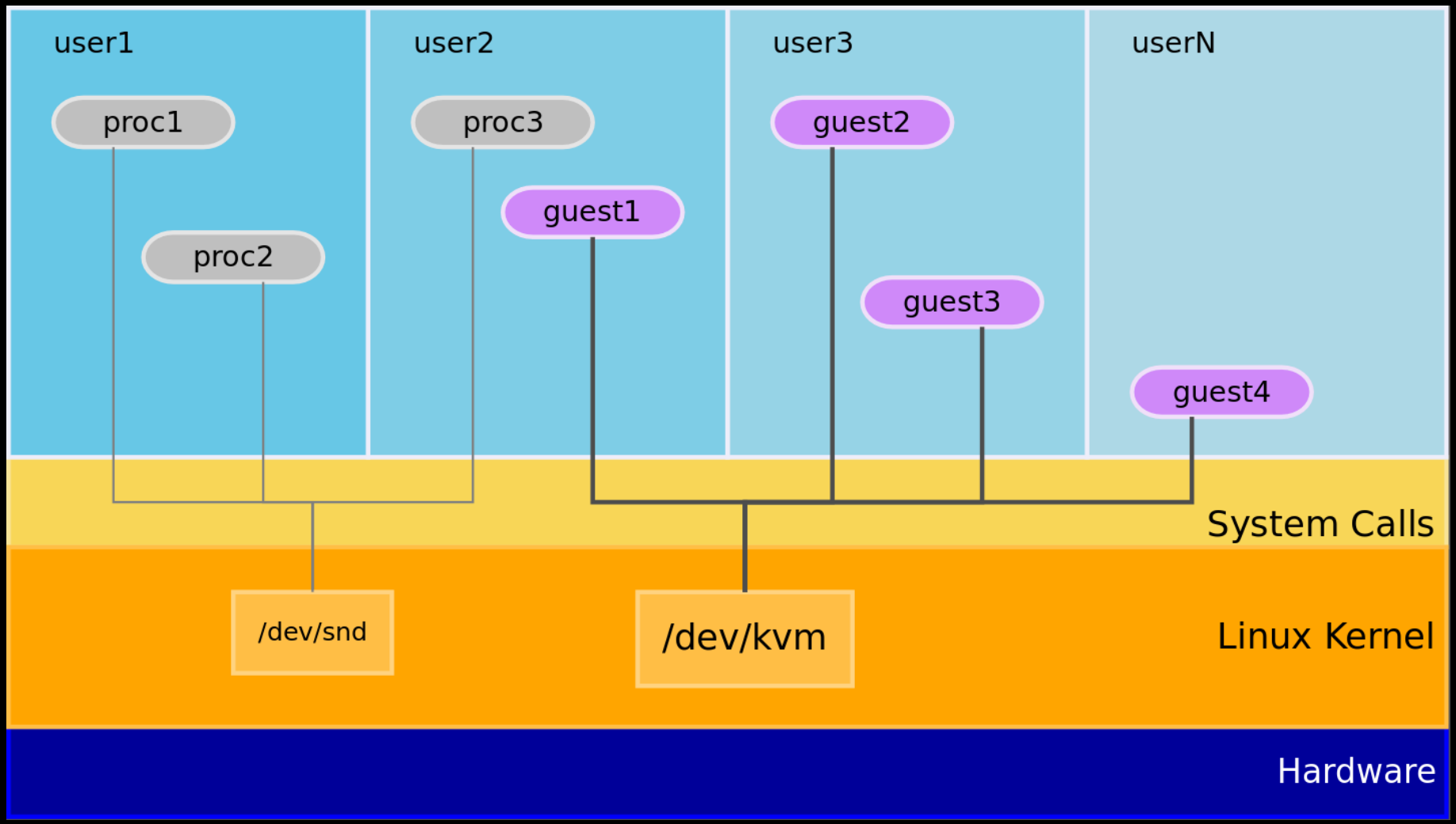
Rede virtual

SELinux + sVirt

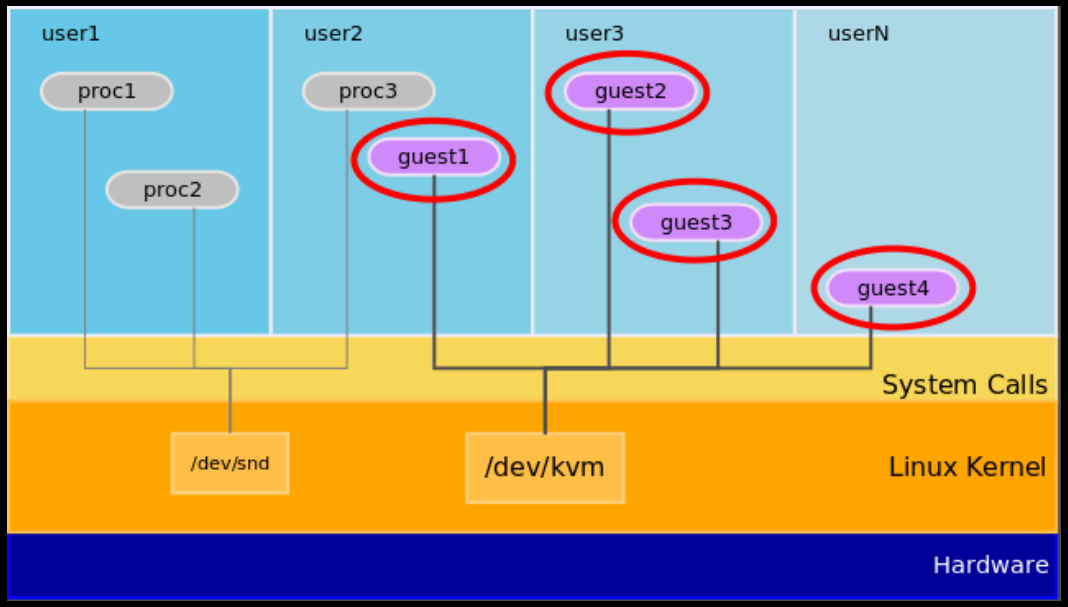
Auditoria

Criptografia em imagens

Arquitetura



Arquitetura



Máquina Virtual KVM

=

processo

(como qualquer outro)

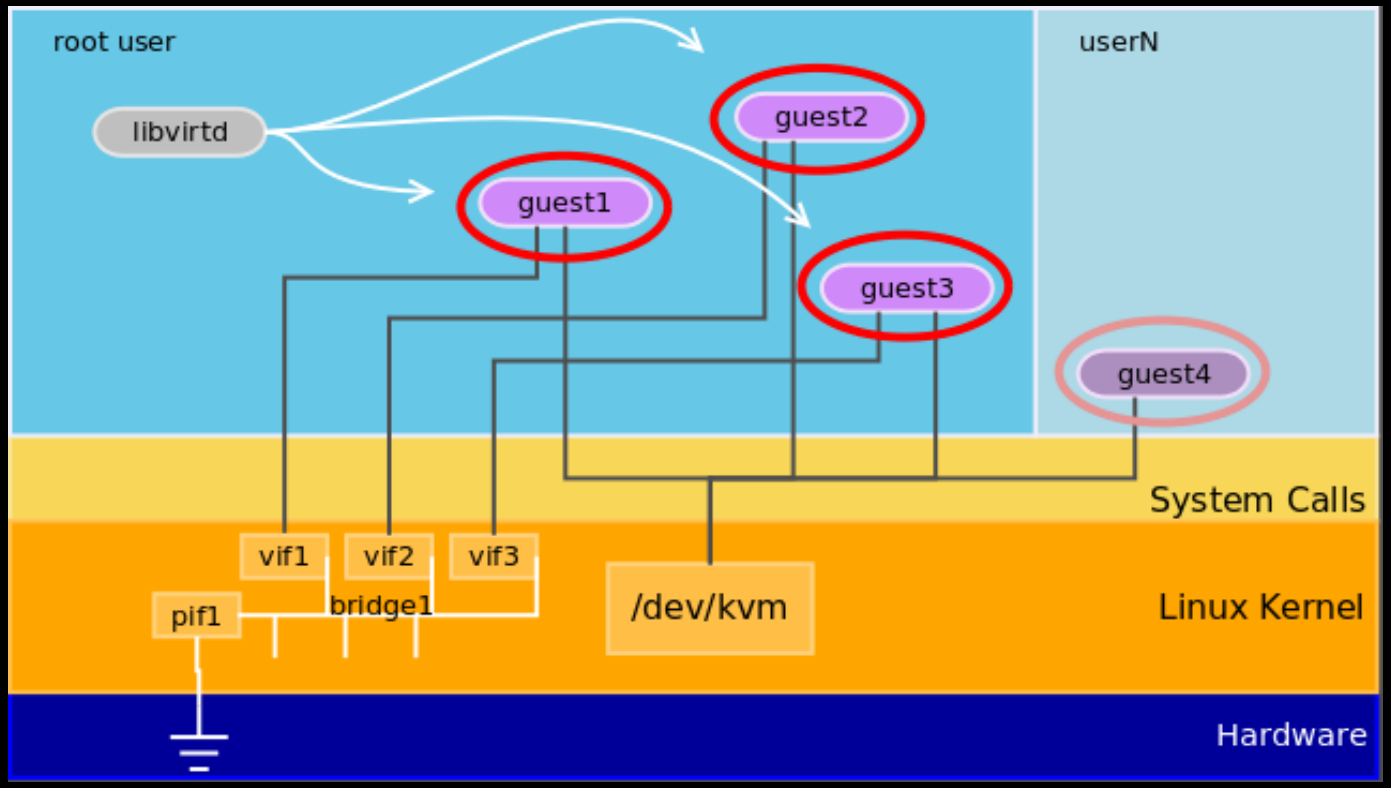
Separação de processos = Qemu + kernel

Arquitetura Modular:

- ✅ De acordo com *filosofia* Open-source/Linux
- ⚠ Mais pontos de “interesse”
- ✅ Re-uso de interfaces conhecidas e testadas

Solução **Libvirt**: simplificar, estabilizar interfaces ✅ ✅ ✅ ✅ …

Arquitetura

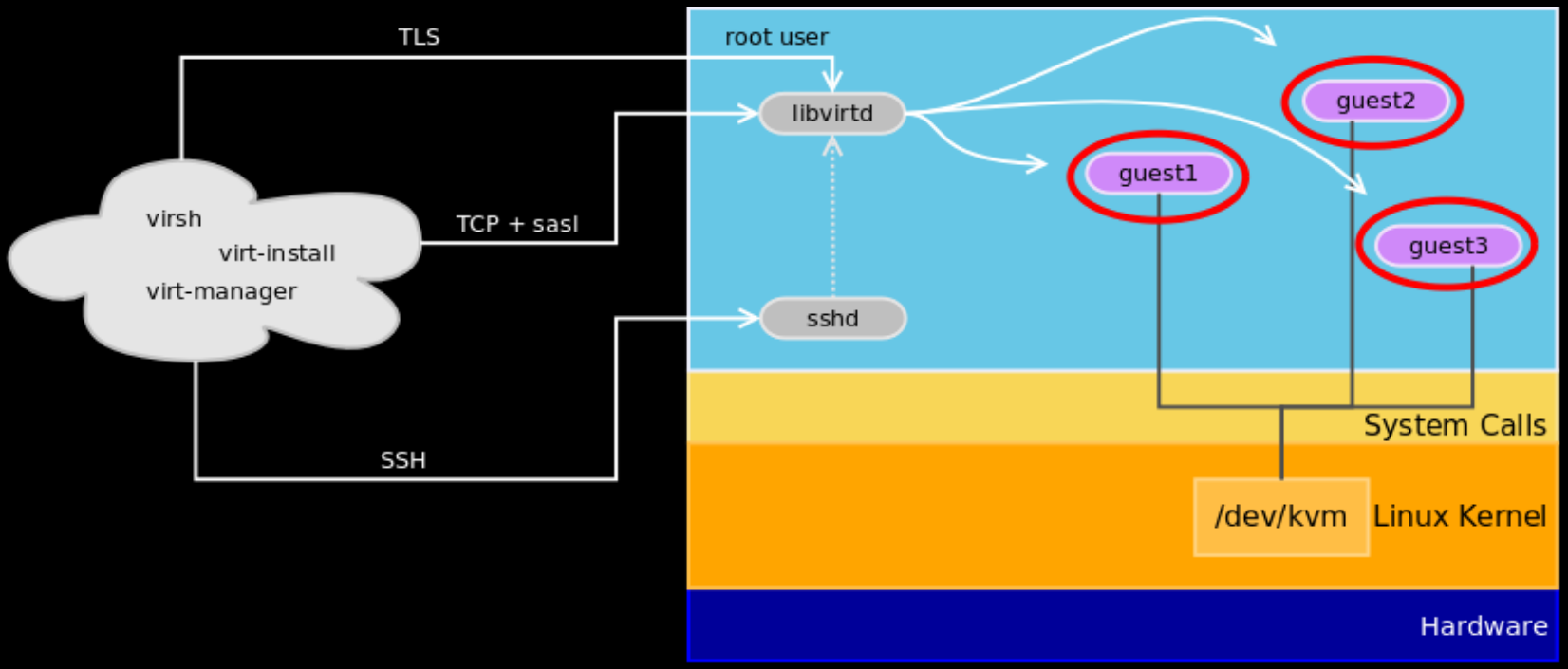


Na prática = Guests rodando como *root*

- Motivo: uso eficiente de Interfaces de rede virtuais

Gerência através de Libvirt

Gerenciamento

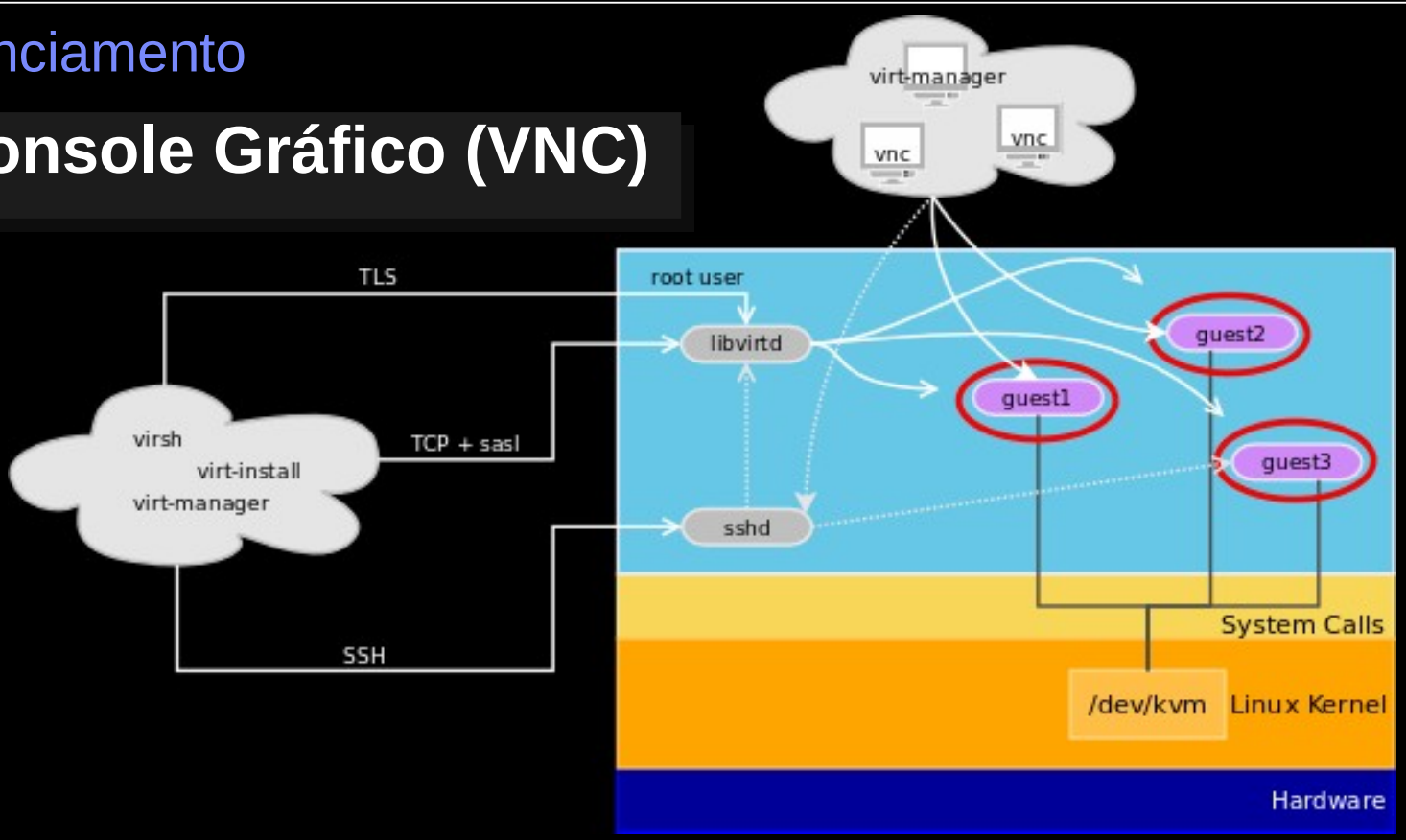


Local ou túnel SSH – autenticação UNIX

TCP + sasl – DB próprio para credenciais

TLS – Autenticação via certificado digital

Console Gráfico (VNC)



TCP simples – sem autenticação ou criptografia

VNC over TLS – certificado digital

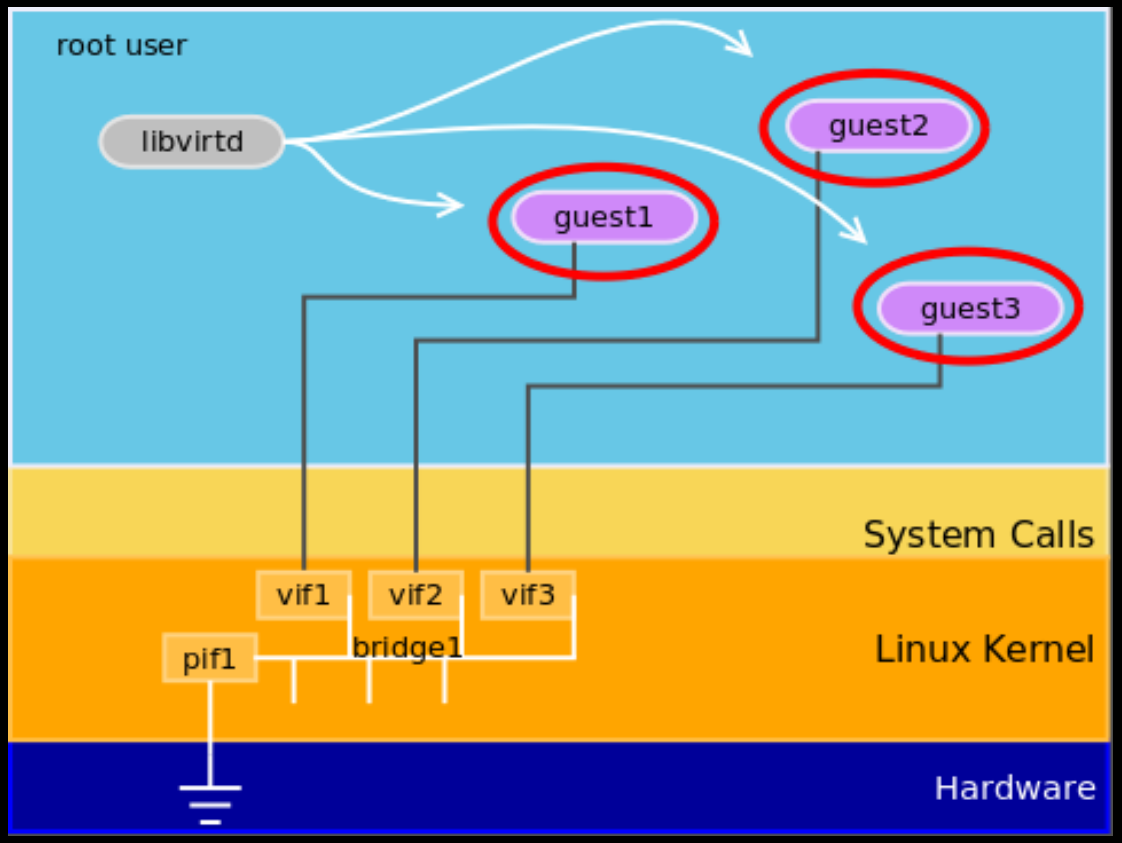
Gerenciamento

Pontos importantes:

- Rede de gerência isolada, firewall
- Console gráfico VNC
  - Range de portas dinâmico
  - Conexão local, autenticação fraca
  - Opção: VNC-over-TLS
- Uso de credenciais *root* com libvirtd
  - Exemplo: gerenciamento via ssh
  - Opções: credenciais SASL, certificados TLS
    - ou AMBOS!

Rede Virtual

Bridge Simples



Link-layer (OSI layer 2) – encaminhamento de *frames*

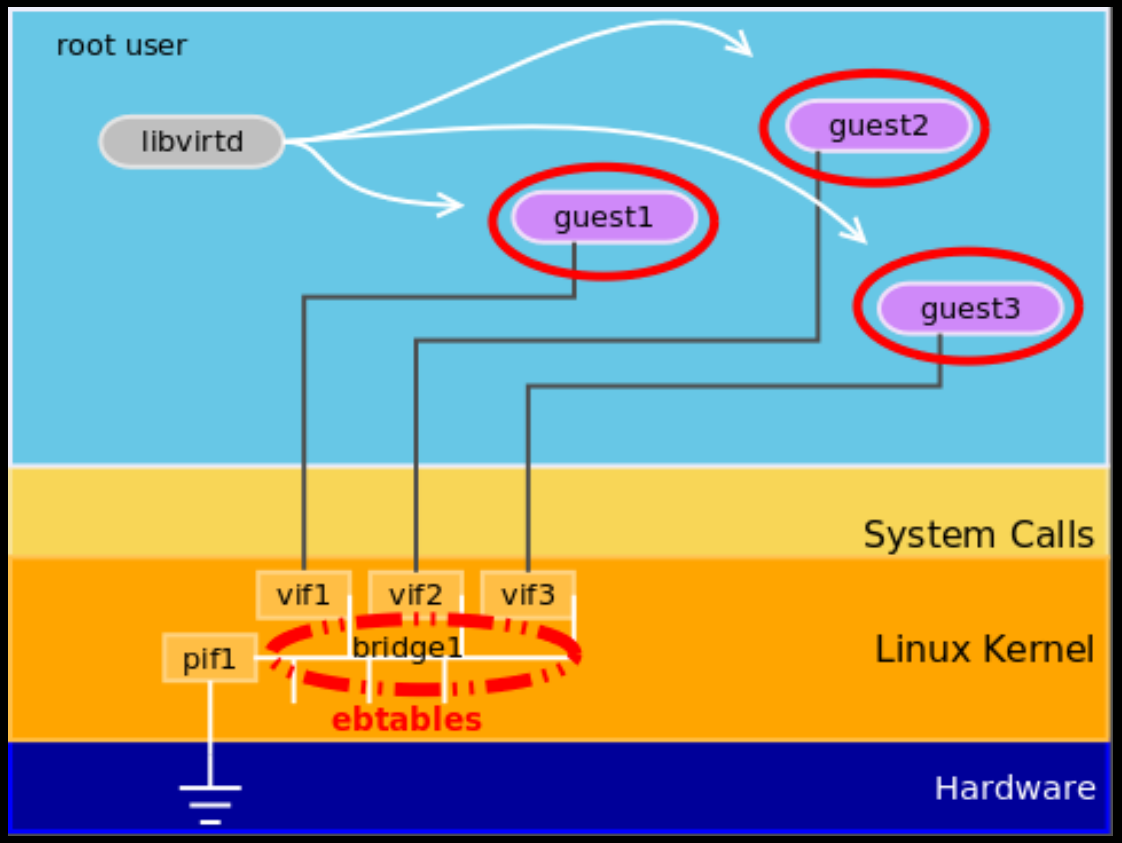
Uso genérico

Riscos:

- ARP spoofing
- MAC flooding

Rede Virtual

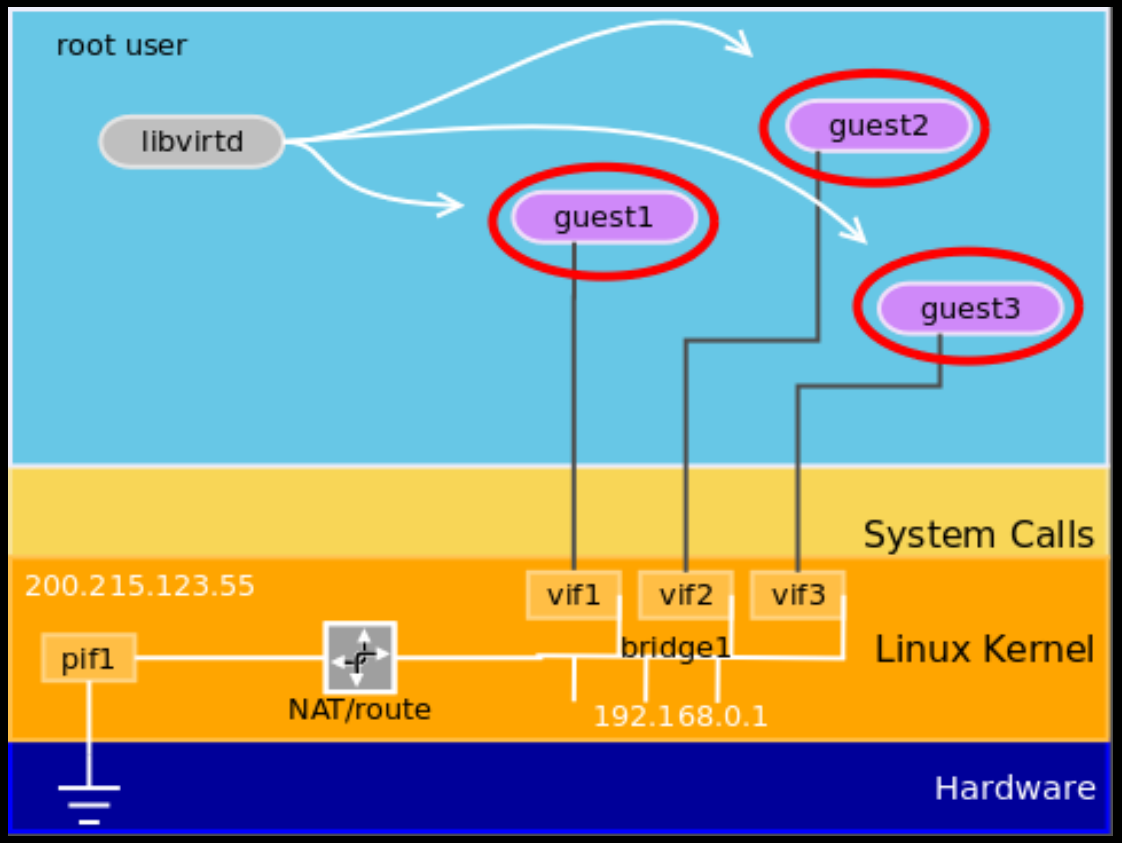
Bridge
Simples
+
ebtables




```
# ebtables -P FORWARD DROP
# ebtables -s 54:52:00:11:22:33 -i vif1 -j ACCEPT
# ebtables -o vif+ -j ACCEPT
```

Rede Virtual

NAT
ou
Routing



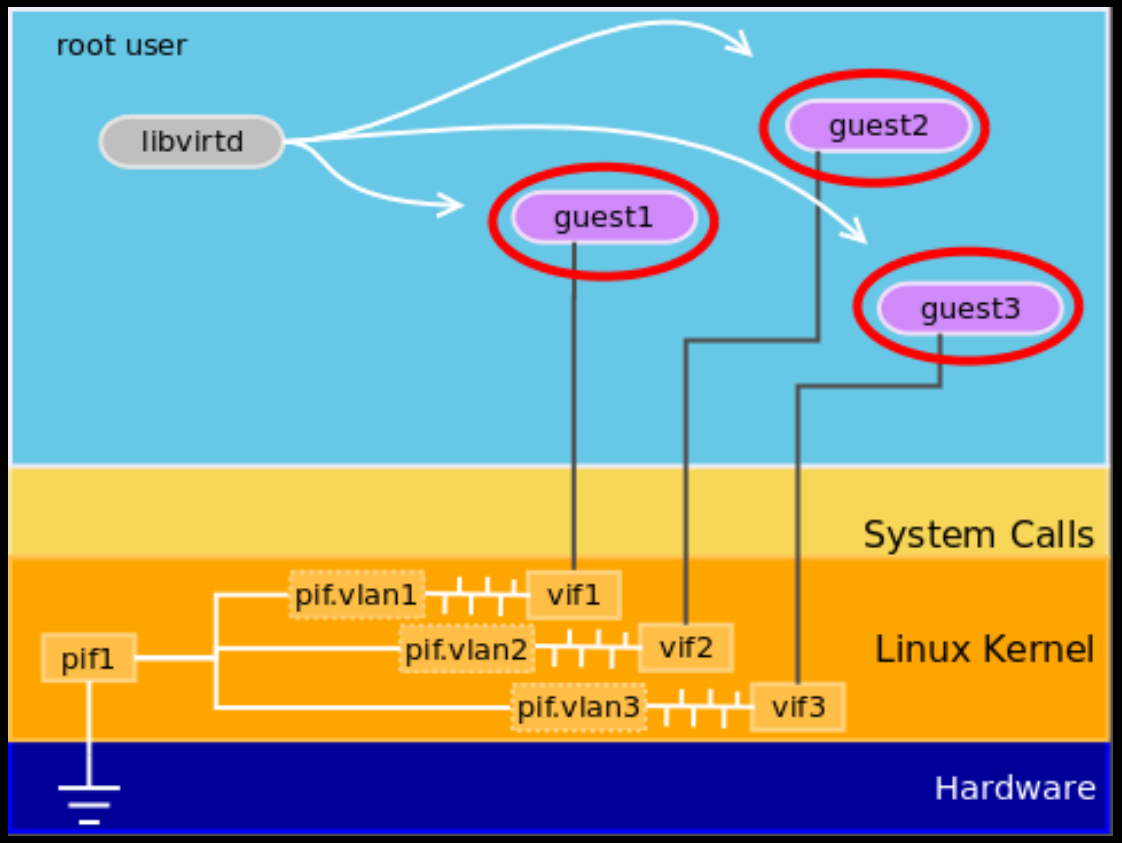
Network-layer (OSI layer 3) – encaminhamento de *pacotes*

+Endereços IP

+bridge

Rede Virtual

VLAN 802.1q



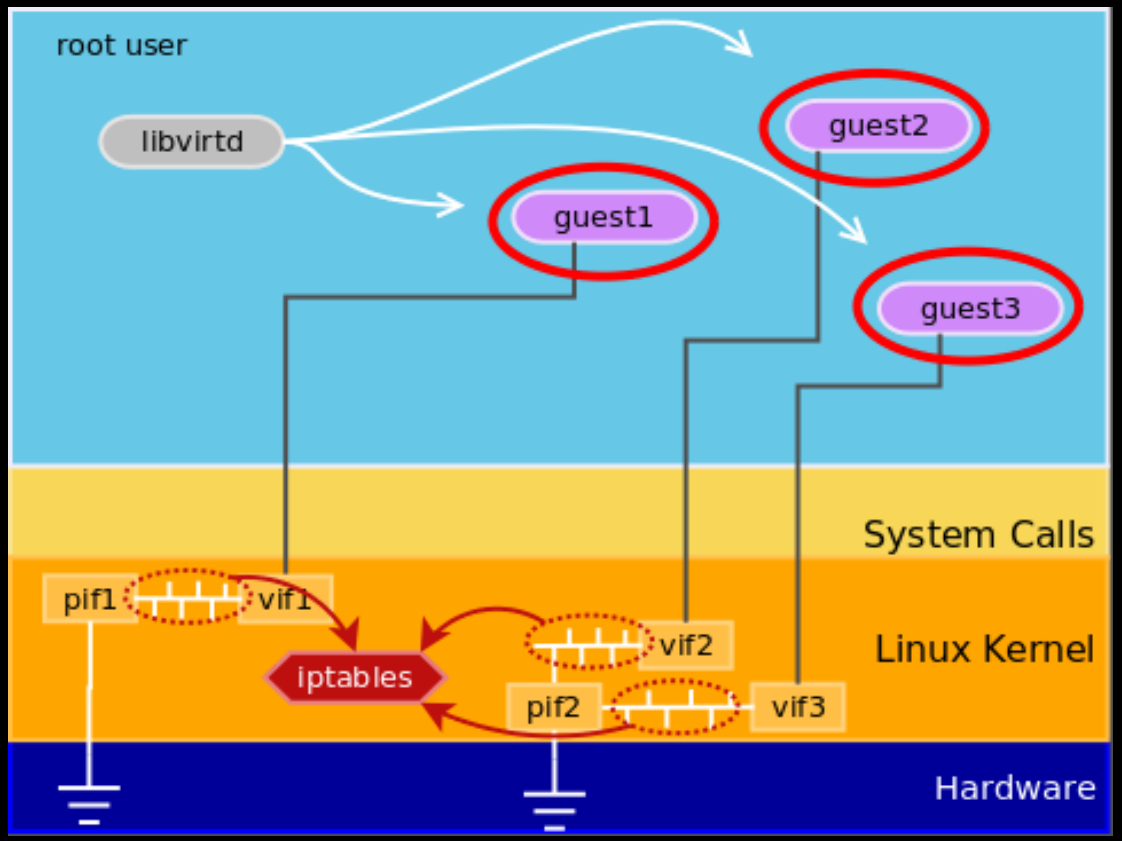
sub-interfaces + bridge

vconfig

Recomendação: 1 bridge por VLAN ID

Rede Virtual

Atenção:
iptables em Bridges

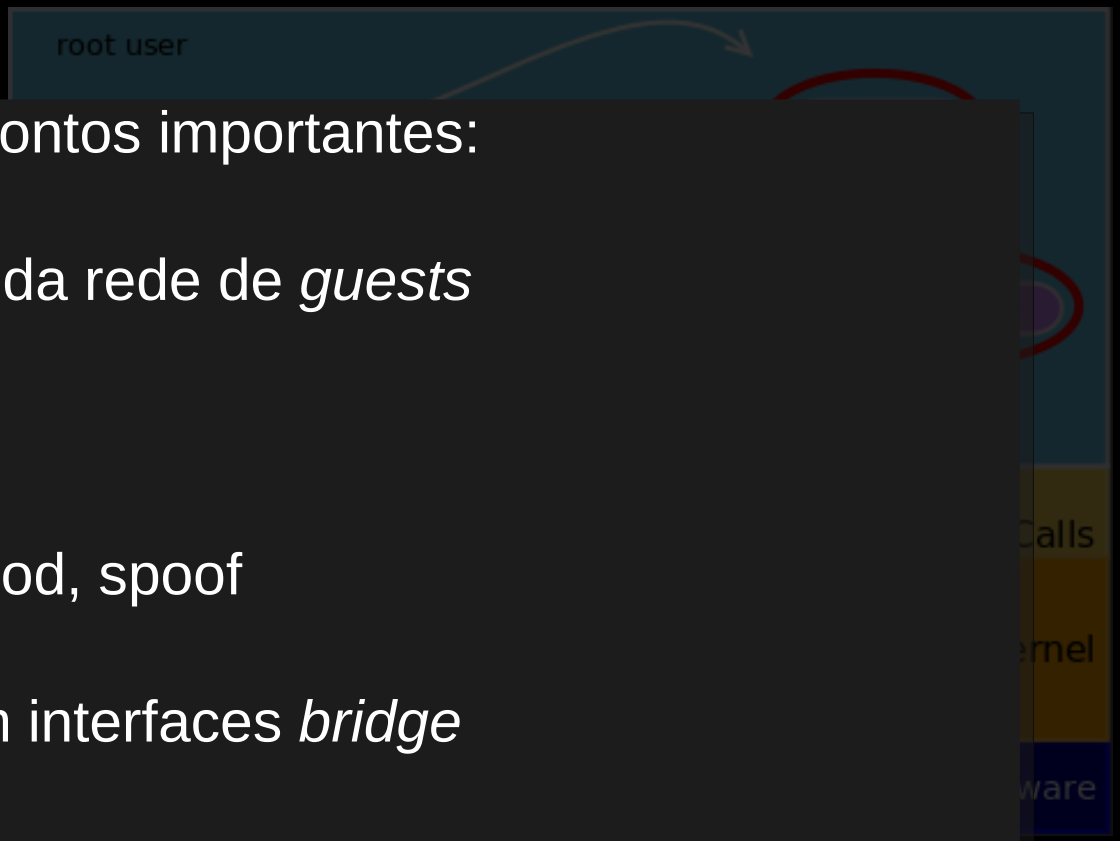


Contexto global em certos módulos como *conntrack*

/etc/sysctl.conf:

```
net.bridge.bridge-nf-call-ip6tables = 0
net.bridge.bridge-nf-call-iptables = 0
net.bridge.bridge-nf-call-arptables = 0
```

Rede Virtual



Pontos importantes:

Configuração estática da rede de *guests*
- MAC fixo

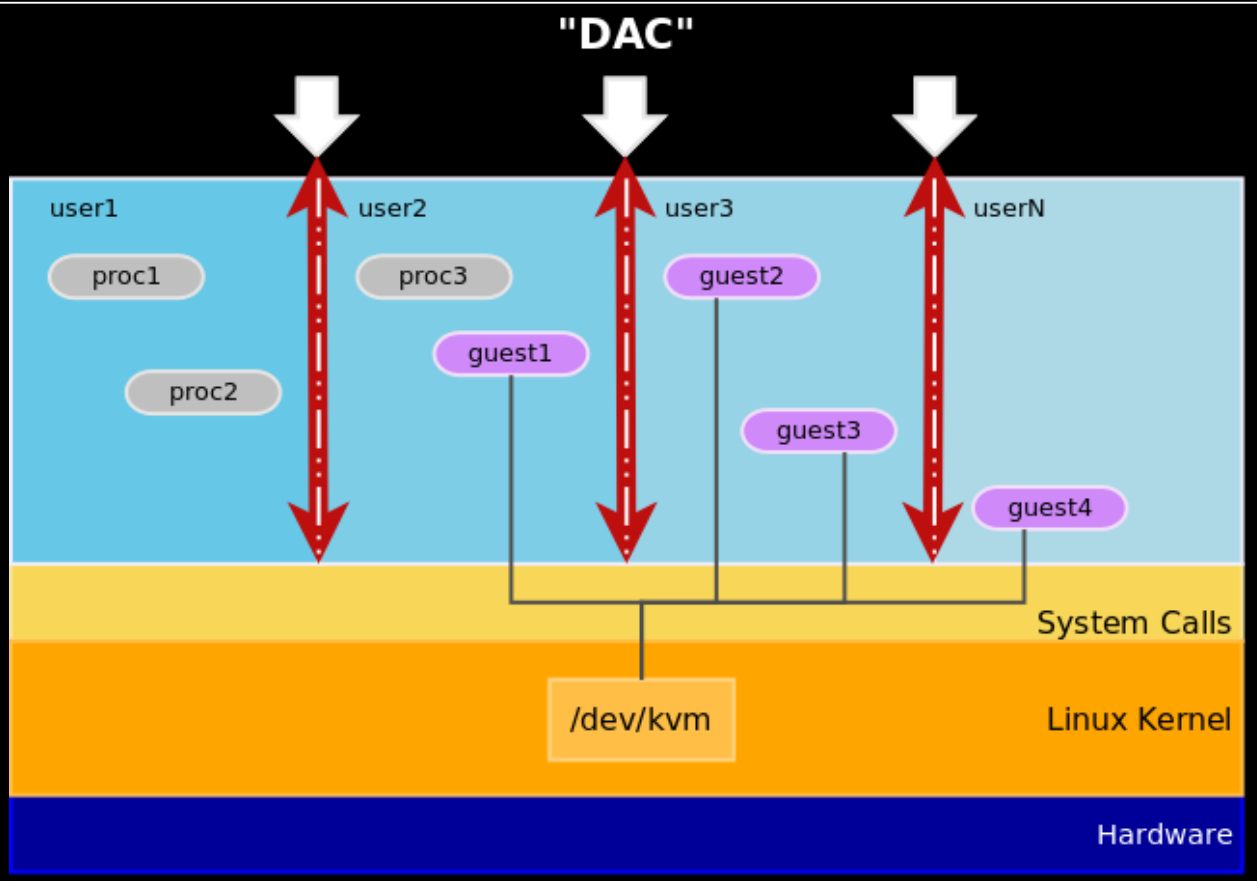
ebtables
- Proteção contra flood, spoof

Desabilitar iptables em interfaces *bridge*

VLANs 802.1q = simplificação

Filtragem avançada – configuração no *host*
- ou use *libvirt* > 0.8.2 (suporte a filtros de rede)

Antes do **SELinux**



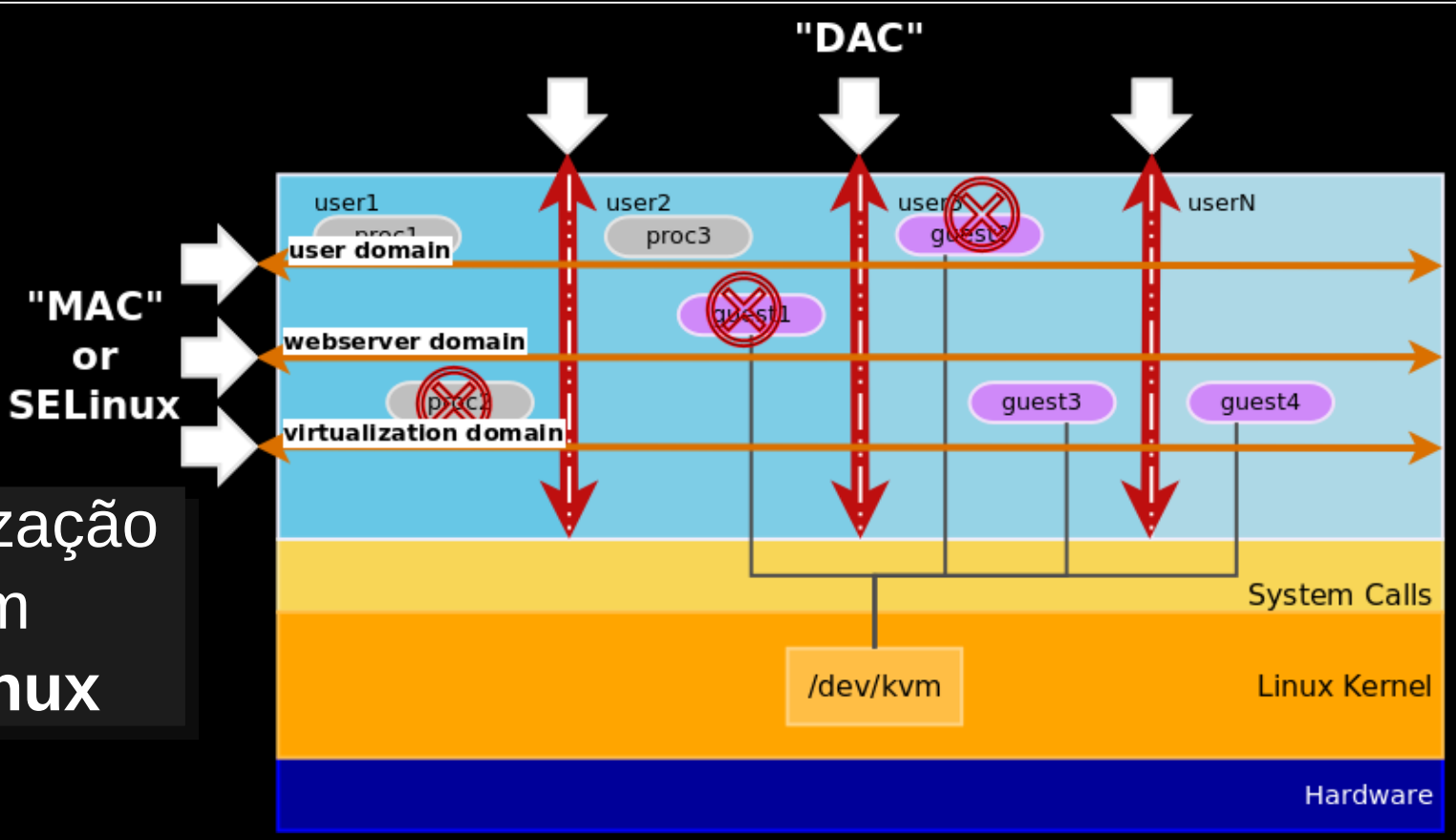
DAC = Discretionary Access Control

Permissões padrões UNIX (user/group/other, read/write/execute)

Usuário controla o acesso aos seus objetos

O mesmo vale para código malicioso ou hackers

SELinux

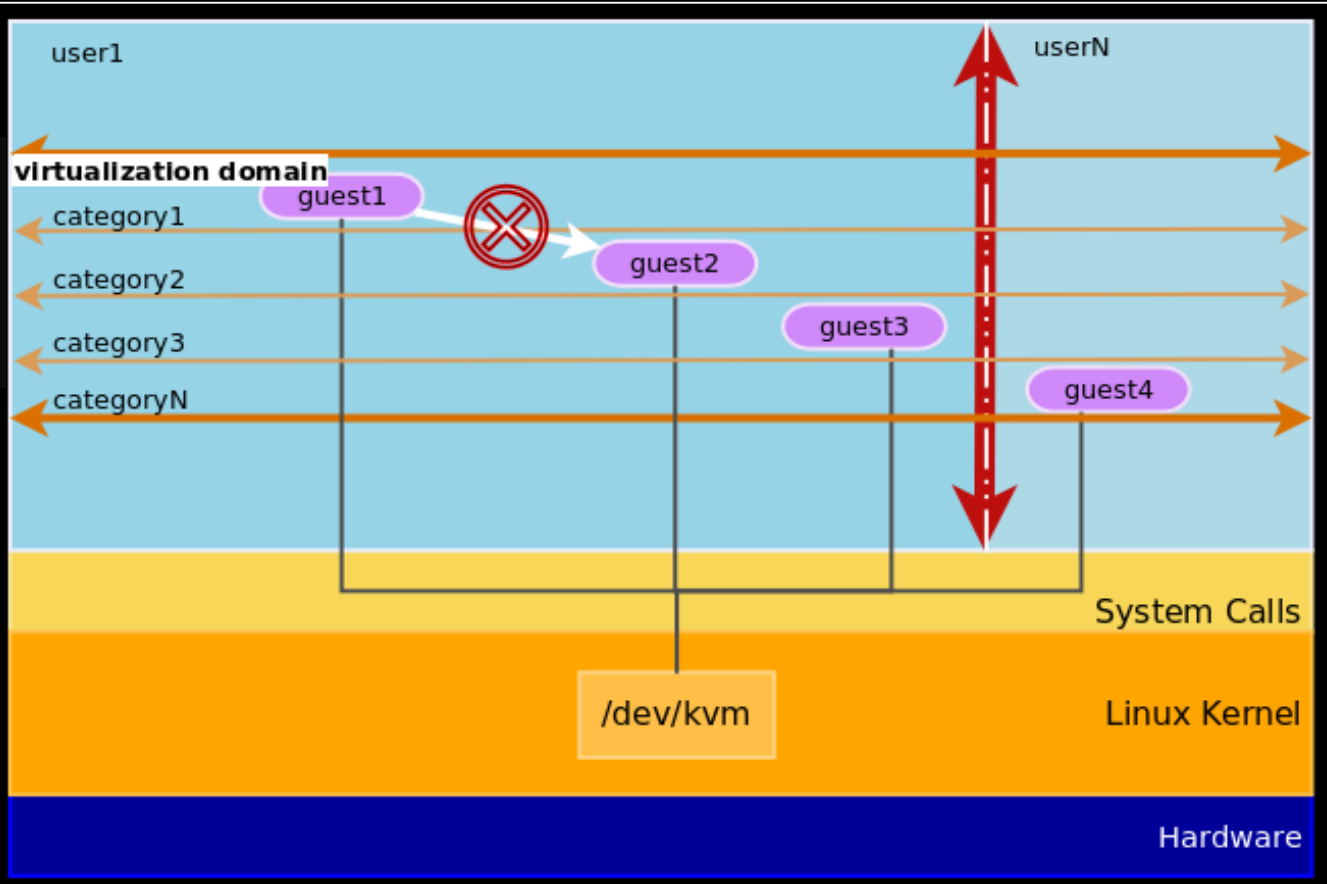


Virtualização com **SELinux**

MAC = Mandatory Access Control

- Permissões definidas por políticas do sistema
- Separação em *domínios*
  - todas as interações explicitamente definidas
- Processos com domínios próprios

Virtualização com **sVirt**



SELinux: cada domínio pode ser quebrado em **categorias** (MCS)

- Cada *guest* roda numa categoria diferente
- Política proíbe qualquer interação entre categorias
- **sVirt** = política **SELinux** + driver **libvirt**

Auditoria

- **Motivação:**
  - Virtualização = atividade relativamente contida
  - Administração requer *root*

- ***Linux Audit* :**

  Bootloader:

```
root (hd0,1)
kernel /vmlinuz-el5.x86_64 ro root=/dev/sda1 rhgb audit=1
initrd /initramfs-el5.x86_64.img
```

```
# chkconfig auditd on
```

`/etc/audit/audit.rules`:

  - Auditar mudanças na configuração
  - Auditar interações *não usuais*

Auditoria

audit.rules: **exemplos**

```
# Acesso chave privada TLS
-a exit,always \
 -F path=/etc/pki/libvirt/private/serverkey.pem \
 -F subj_type!=virtd_t

# Mudanças em imagens
-a exit,always -F obj_type=virt_image_t \
 -F perm=wa -F subj_type!=qemu_t

# Interações do Qemu com outros domínios
-a exit,always -F arch=b64 -S all -F perm=wax \
 -F subj_type=qemu_t -F obj_type!=qemu_t
```

Mais exemplos, cenários mais detalhados:

Blueprint: "*Securing KVM guests and the host system*"

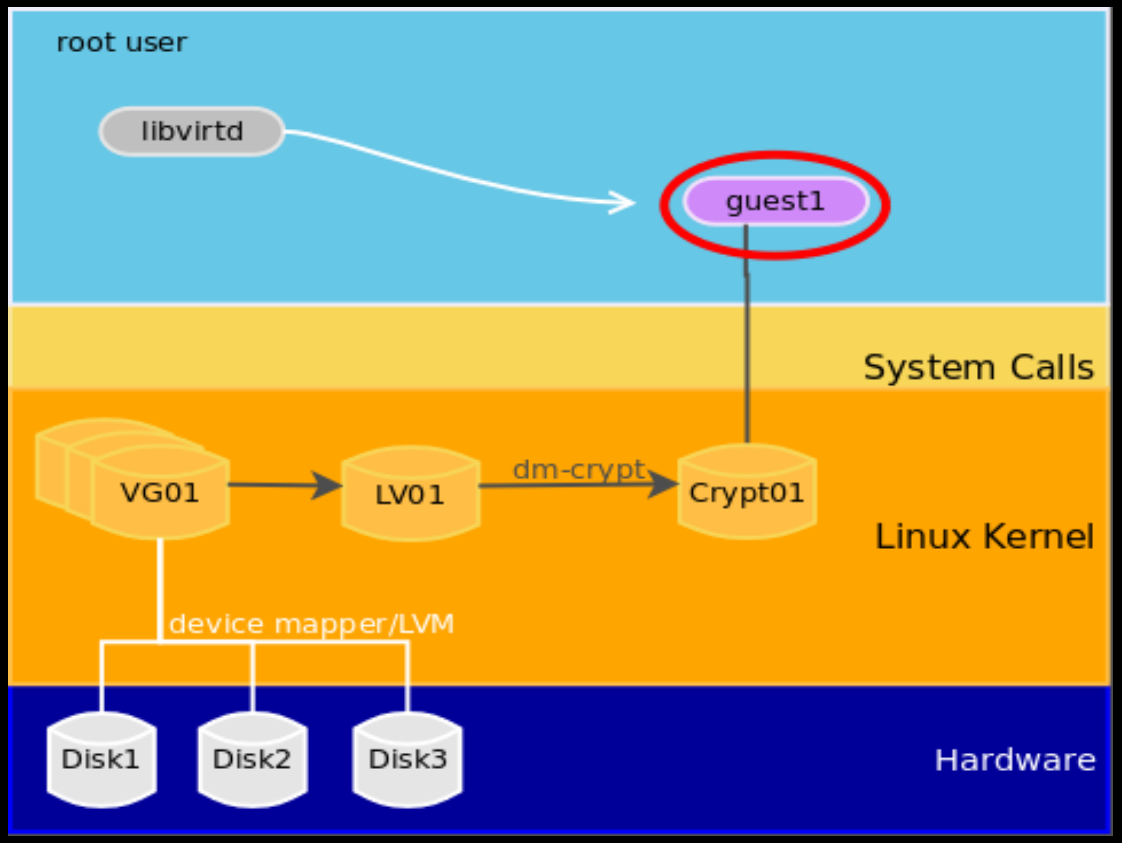
http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/topic/liaai/kvmsec/kvmsecstart.htm

ou

http://blog.klauskiwi.com/

Cryptografia em imagens

dm-crypt
e
qcow2



- Garantir proteção de *dados em repouso* (*data-at-rest*)
- dm-crypt = cifragem em dispositivo de bloco (kernel)
  - Transparente para o KVM (imagem *raw*)

```
# cryptsetup
```

- qcow2 = formato suporta crypto, sparse files, libvirt ...

Dúvidas?

Klaus Heinrich Kiwi
klaus@klauskiwi.com

Apresentação disponível em:
http://blog.klauskiwi.com